# Computing with sequents and diagrams in classical logic - calculi *X, dX and ©X

Dragisa Zunic

N° d'ordre: 447
N° attribue par la bibliothèque: 07ENSL0 447

# THÈSE

*en vue d'obtention le grade de*

Docteur de l'École Normale Supérieure de Lyon

spécialité : Informatique

*Laboratoire de l'Informatique du Parallélisme*
*École Doctorale de Mathématiques et Informatique Fondamentale*

*présenté et soutenue publiquement le* 21 decembre 2007 *par monsieur*

Dragiša Žunić

## Computing with Sequents and Diagrams in Classical Logic - Calculi ${}^{*}\mathcal{X}$, ${}^{d}\mathcal{X}$ and ${}^{©}\mathcal{X}$

Directeur de theèse : *Monsieur* Pierre Lescanne

Après avis de : *Monsieur* Steffen van Bakel
*Monsieur* Hugo Herbelin

Devant la commission d'examen formèe de :

| | |
|---|---|
| *Monsieur* Steffen van Bakel | Membre/Rapporteur |
| *Monsieur* Hugo Herbelin | Membre/Rapporteur |
| *Madam* Delia Kesner | Membre |
| *Monsieur* Yves Lafont | Membre |
| *Monsieur* Pierre Lescanne | Membre |
| *Monsieur* Christian Urban | Membre |

# Contents

# Abstract

This Ph.D. thesis addresses the problem of giving computational interpretation to proofs in classical logic. As such, it presents three calculi reflecting different approaches in the study of this area.

The thesis consists of three parts.

The first part introduces the ${}^{*}\mathcal{X}$ calculus, whose terms represent proofs in the classical sequent calculus, and whose reduction rules capture most of the features of cut-elimination in sequent calculus. This calculus introduces terms which enable explicit implementation of *erasure* and *duplication* and to the best of our knowledge it is the first such calculus for classical logic.

The second part studies the possibility to represent classical computation diagrammatically. We present the ${}^{d}\mathcal{X}$ calculus, the *diagrammatic* calculus for classical logic, whose diagrams originate from ${}^{*}\mathcal{X}$-terms. The principal difference lies in the fact that ${}^{d}\mathcal{X}$ has a higher level of abstraction, capturing the essence of sequent calculus proofs, as well as the essence of classical cut-elimination.

The third part relates the first two. It presents the ${}^{\copyright}\mathcal{X}$ calculus, a one-dimensional counterpart of the diagrammatic calculus. We start from ${}^{*}\mathcal{X}$, where we explicitly identify terms which should be considered the same. These are the terms that code sequent proofs which are equivalent up to permutations of independent inference rules. They also have the same diagrammatic representation. Such identification induces the *congruence relation* on terms. The reduction relation is defined modulo congruence rules, and reduction rules correspond to those of ${}^{d}\mathcal{X}$ calculus.

# Résumé

Cette thèse de doctorat étudie l'interprétation calculatoire des preuves de la logique classique. Elle présente trois calculs reflétant trois approches différentes de la question.

Cette thèse est donc composée de trois parties.

La première partie introduit le ${}^{*}\mathcal{X}$ calcul, dont les termes représentent des preuves dans le calcul des séquents classique. Les règles de réduction du ${}^{*}\mathcal{X}$ calcul capture la plupart des caractéristiques de l'élimination des coupures du calcul des séquents. Ce calcul introduit des termes permettant une implémentation implicite de l'*effacement* et de la *duplication*. Pour autant que nous sachions, c'est le premier tel calcul pour la logique classique.

La deuxième partie étudie la possibilité de représenter les calculs classiques au moyen de diagrammes. Nous présentons le ${}^{d}\mathcal{X}$ calcul, qui est le calcul *diagrammatique* de la logique classique, et dont les diagrammes sont issus des ${}^{*}\mathcal{X}$-termes. La différence principale réside dans le fait que ${}^{d}\mathcal{X}$ fonctionne à un niveau supérieur d'abstraction. Il capture l'essence des preuves du calcul des séquents ainsi que l'essence de l'élimination classique des coupures.

La troisième partie relie les deux premières. Elle présente le ${}^{\copyright}\mathcal{X}$ calcul qui est une version unidimensionnelle du calcul par diagramme. Nous commençons par le ${}^{*}\mathcal{X}$, où nous identifions explicitement les termes qui doivent l'être. Ceux-ci sont les termes qui encodent les preuves des séquents qui sont équivalentes modulo permutation de règles d'inférence indépendantes. Ces termes ont également la même représentation par diagramme. Une telle identification induit une *relation de congruence* sur les termes. La relation de réduction est définie modulo la congruence, et les règles de réduction correspondent à celle du ${}^{d}\mathcal{X}$ calcul.

# Rezime

Ова докторска теза бави се проблемом давања нових рачунарских интерпретација за доказе у класичној логици. Као таква, представља три нова формална рачуна који осликавају различите приступе у студирању ове области, а при томе чине једну целину.

Теза се састоји из три дела.

Први део уводи ${}^{*}\mathcal{X}$ рачун, калкулус чији терми представљају доказе у класичном секвентном рачуну, док редукцијска правила представљају понашање доказа приликом трансформације познате под називом - елиминација резова. Овај рачун уводи нове терме који служе за имплементацију операција брисања и дуплирања и, колико нам је познато, ово је први такав рачун за класичну логику представљен до сада.

Други део студира могућност увођења диаграматске интерпретације за рачунање које одговара класичној логици. Предсављамо ${}^{d}\mathcal{X}$ рачун, диаграматки рачун за класичну логику, чији диаграми воде порекло од ${}^{*}\mathcal{X}$-терма. Основна разлика лежи у чињеници да је ${}^{d}\mathcal{X}$ рачун вишег нивоа апстракције, хватајући управо суштински део, како класичних доказа, тако и њихове трансформације путем елиминисања резова.

Трећи део представља ${}^{\copyright}\mathcal{X}$ рачун, једнодимензионални парњак диаграматског рачуна. Почињемо од ${}^{*}\mathcal{X}$-а, где вршимо експлицитну идентификацију терма које би требало сматрати суштински једнаким, пошто одговарају доказима у секвентном рачуну који су еквивалентни до на пермутацију независних правила извођења. Релација редукције је дефинисана модуло конгруентни терми, а редукцијска правила одговарају онима у ${}^{d}\mathcal{X}$ рачуну.

# Acknowledgements

First of all I thank my thesis adviser Pierre Lescanne, for his patience and generous support and for sharing his knowledge with me. For his always positive energy which helped me try use the chance I had been given.

I thank Steffen van Bakel and Hugo Herbelin for accepting to be the referees for my thesis and for the valuable comments on how to improve it. Delia Kesner, Yves Lafont and Christian Urban for accepting to be the members of the thesis committee, and for their valuable comments.

I thank Silvia (Silvia Ghilezan) for giving me the chance to come to France. For introducing me to logic and computation in an interesting way and for her support. My niece Anastazia for entertaining me here for a few months.

My brother Joviša, for always believing in me and initiating me into this this world of maths when I was still young. I guess it was not a mistake. For inspiring me with his talent you don't see every day.

I thank my parents Sara and Dragomir to whom I can not express the gratitude with words.

I thank my girlfriend Iona for her love and support and for proving that London and Lyon are not so far away.

Members of the "equipe Plume" Daniel, Philippe, Jean, Tom, Stéphane, Damien, Aurelien, Romain, Samuel and Sébastien for always enjoyable time.

Oriol, Shinsuke, Pushpinder, David, Dino, Myriam and the rest of the student crew from and around the ENS.

Silvia Likavec for welcoming me in Lyon and introducing me to many things in a few months, from how to open the main door to how to write a thesis.

All my teachers and professors during more than 20 years.

Finally I thank everyone, and there are many people, with whom I shared the good moments in Yugoslavia, Serbia and France.

# Chapter 1

# Introduction

The history of attempts to understand and model the way we reason is over two thousand years old. One of the most significant fields of theory of reasoning is known as *classical logic*, initiated by a Greek philosopher Aristotle who lived in the fourth century BC. He formulated the notion of logical consequence and the law of non-contradiction similarly to their modern definition [AriBC].[1] Gottlob Frege (1848-1925) is considered to be the first to *formalize* the notion of logical consequence, using only purely syntactic manipulations. The style he used is today known as *Hilbert-style* after David Hilbert (1862-1943). Then it was Gerhard Gentzen (1909-1945) who introduced two new formalisms, namely the *sequent calculus* and the *natural deduction calculus* which can be used to represent classical logic. His concept of *cut-elimination* is still one of the central points of investigations in proof theory.

The fundamental connection between formal proof theory and programming language theory is known as the *Curry-Howard correspondence*, in tribute to Haskell B. Curry and William A. Howard, who were among the first to observe it [CF58, How80]. Originally discovered as a relation between the intuitionistic logic (represented in natural deduction formalism) and simply typed $\lambda$-calculus, it successfully relates logical concepts *proof/normalization/proposition* with computational concepts *term/reduction/type*. This type of correspondence was later studied for other logics, and

[1] There are also much older documents indirectly related to logic, such as the "Code of Hammurabi", a Babylonian law which dates back to 1760 BC. It is a collection of rules, each rule presented by one sentence, where all sentences have the same grammatical structure "If *crime*, then *punishment*". The Code of Hammurabi is related to an older Sumerian document known as the "Code of Ur-Nammu" which has the same form and dates back to 2100 BC, [AS03].

therefore we have a notion of a Curry-Howard *paradigm*.

For a long time, it was considered not possible to include classical logic in to the Curry-Howard paradigm. However, around 1990 Tim Griffin made an important discovery, which showed that classical logic has something to offer in this sense. In his seminal work [Gri90], he noted the connection between functional programming languages with control operators and classical logic, which opened the door for the continuation of research in the field.

Classical computation was first studied in the framework of natural deduction. One of the first calculi to be designed is $\lambda\mu$-calculus of Parigot [Par92], in 1992. There are certain technical reasons which make natural deduction less suitable then sequent calculus for the study of classical logic, and since then several calculi based on the sequent calculus formalism have been designed. For example, one of the first was $\bar{\lambda}\mu\tilde{\mu}$-calculus of Curien and Herbelin [CH00], a calculus which describes classical computation in only several reduction rules, and exhibits various symmetries and dualities. It corresponds to a formulation of sequent calculus where sometimes one formula in the sequent is distinguishable, which still has a flavor of intuitionistic logic.

This thesis' research started by studying the $\mathcal{X}$ calculus [vBLL05], which is a somewhat new term-language, directly associated to sequent calculus for classical logic. It uses the basic concept of *name* instead of that of variable, and captures the structure of sequent proofs as well as the process of cut-elimination. The $\mathcal{X}$ calculus as a model is very close to Bierman and Urban's term-notation for classical sequents [UB01a], studied in great detail in [Urb00]. The properties and results presented in the framework of $\mathcal{X}$ calculus have been a starting point for this work.

Another source of inspiration comes from intuitionistic logic. The computational aspects of erasure and duplication of explicit substitution were studied more in the intuitionistic framework. Although there are numerous works in this field, the main influence comes from Kesner and Lengrand's design of $\lambda$lxr-calculus [KL07], which introduces terms for erasure and duplication of explicit substitution, without loosing good properties of its underlying calculi.

As a result of these investigations, combining the ideas from $\lambda$lxr and $\mathcal{X}$, we developed three new calculi that reflect different views on the classical computation. Therefore the thesis itself consists of three parts, each part presenting one calculus.

*The first part* presents the ${}^{*}\mathcal{X}$ calculus, which captures the structure of sequent proofs, even more comprehensively than it was the case with $\mathcal{X}$ calculus. Namely, ${}^{*}\mathcal{X}$ has been designed to provide a correspondence "à la"

Curry-Howard with Gentzen's sequent system $G_1$ for classical logic. The specific feature of the system $G_1$ is explicit presence of the structural inference rules *weakening* and *contraction*. The design of the calculus reveals the role of the structural rules (weakening and contraction) in the process of proof-transformation, i.e., on the computational side they appear as terms called *eraser* and *duplicator*, respectively, as they are used to implement erasure and duplication. A computation in ${}^{*}\mathcal{X}$ corresponds to a cut-elimination in the sequent calculus. To the best of our knowledge, it is the first calculus for classical logic which explicitly implements erasure and duplication. The fact that ${}^{*}\mathcal{X}$ carries even higher level of details than $\mathcal{X}$ is justified since it provides us with detail on how exactly erasure and duplication work. The existing classical calculi are not entirely explicit about the operations of erasure and duplication of terms, although these operations are important in both, the theory and the implementation. A second justification is that by having all the detail explicit in the calculus, we could catch the syntax and the rules of proofs seen as diagrams, which leads to another calculus we designed.

*The second part* presents the *diagrammatic calculus* for classical logic, called ${}^{d}\mathcal{X}$. In some sense this is the natural continuation of the first part; thanks to linearity and the presence of erasers and duplicators, ${}^{*}\mathcal{X}$-terms can be seen as two dimensional diagrams. This means that we define a diagrammatic syntax and the reduction relation is given by the rewriting rules over those diagrams. The static aspects of this calculus have recently been defined by Robinson [Rob03] as proof-nets for classical logic, but the dynamic aspects (computation) have not yet been presented. The diagrams have been inspired by Girard's proof-nets for linear logic [Gir87], although there is a difference in a sense that we search for a system which is in direct relation with standard sequent system for classical logic. ${}^{d}\mathcal{X}$ is a higher level abstraction calculus, as it abstracts away from unessential details which are unavoidable in a formalism like a sequent calculus. Although it might seem unexpected, it was necessary to dive into detail (${}^{*}\mathcal{X}$) to be able to derive a more abstract model (${}^{d}\mathcal{X}$), which captures only the essential features of the classical computation. If we also have in mind the importance of visual images to human cognitive activities, it is not difficult to motivate the study of ${}^{d}\mathcal{X}$. The calculi ${}^{*}\mathcal{X}$ and ${}^{d}\mathcal{X}$ are not in one-to-one correspondence, since more than one term can correspond to one diagram.

*The third part* deepens the relation between one-dimensional, namely ${}^{*}\mathcal{X}$, and two-dimensional, namely ${}^{d}\mathcal{X}$ calculi. And the result is the ${}^{\copyright}\mathcal{X}$-calculus, which is a one-dimensional counterpart of the diagrammatic calculus. It is obtained from ${}^{*}\mathcal{X}$ by stating which (syntactically different) terms should be

considered the same. This induced a *congruence relation* on terms, and the reduction relation in ${}^{\copyright}\mathcal{X}$ is defined modulo the congruence rules. Notice that this is an explicit attempt to say *which* syntactically different terms (sequent proofs) are, in fact, essentially the same. This would have been difficult to infer directly, using sequent calculus formalism. But having at hand both terms of ${}^{*}\mathcal{X}$ and diagrams of ${}^{d}\mathcal{X}$, yields the result naturally. Thus this thesis also addresses the very old problem of proof identity, caused by the lack of understanding of the essence of a proof, which is in turn caused by the "bureaucracy" involved in the syntactic presentation of proofs (see for example [Str06]).

The three calculi are non-deterministic and non-confluent. The reduction rules satisfy *free names* preservation (interface preservation), *type* preservation (computation can be seen as proof-transformation). The computation also preserves the *linearity* of terms.

So one of the main aspects of this work is revisiting the proof theory of classical logic, where proofs are identified up to trivial rule permutations.

## 1.1 Outline of the thesis

This thesis is composed of three parts. Each part presents one formal language through which we study different aspects of classical computation.

- Part I presents a way to implement both, explicit erasure and duplication in the framework of classical computation. We study this by introducing the ${}^{*}\mathcal{X}$ calculus.
  - In Chapter 2 we give an introduction to the ${}^{*}\mathcal{X}$ calculus. We also recall the basic philosophy behind the its predecessors, $\mathcal{X}$ and the $\lambda$lxr-calculus.
  - Chapter 3 reviews Gentzen's sequent calculus LK, with its basic variants with respect to the treatment of structural rules.
  - In Chapter 4 we present in more detail the related calculi, namely the $\mathcal{X}$ calculus, the $\lambda$lxr-calculus and $\bar{\lambda}\mu\tilde{\mu}$-calculus.
  - Chapter 5 presents the ${}^{*}\mathcal{X}$ calculus. We present the syntax, the reduction rules and the basic operational properties. Then we define the typing rules, and we show that the computation preserves types. The chapter ends by showing how we can implement data types in ${}^{*}\mathcal{X}$ calculus.
  - In Chapter 6 we show how we can encode the related calculi in ${}^{*}\mathcal{X}$.

- Part II presents the diagrammatic calculus ${}^{d}\mathcal{X}$. It comes as a result of studying the ${}^{*}\mathcal{X}$ calculus, which has all ingredients for the diagrammatic representation.
  - Chapter 7 gives an introduction to diagrammatic reasoning. We recall some old and some of the recent works.
  - Chapter 8 presents the diagrammatic calculus ${}^{d}\mathcal{X}$, with its syntax and reduction rules. Besides that, we introduce the so-called simplification rules for diagrams and basic operational properties. We end the chapter by defining the type assignment system for the language, with the property of type-preservation.
- Part III shows the relation between one-dimensional ${}^{*}\mathcal{X}$-terms and ${}^{d}\mathcal{X}$-diagrams. Moreover, we relate the computational aspects of the two approaches. The result is the ${}^{\copyright}\mathcal{X}$ calculus, where reducing is defined modulo congruence.
  - Chapter 9 is an introduction to the ${}^{\copyright}\mathcal{X}$ calculus.
  - In Chapter 10 we illustrate the principal differences between one-dimensional and diagrammatic computing.
  - Chapter 11 presents the ${}^{\copyright}\mathcal{X}$ calculus. We introduce the congruence relation on terms. After that, we define reduction rules as reduction modulo congruence rules. We show that propagation rules are not needed. Then we give the basic operational properties and define the typed language. Furthermore, we show that ${}^{\copyright}\mathcal{X}$ and ${}^{d}\mathcal{X}$ are closely related; each reduction step of ${}^{\copyright}\mathcal{X}$ corresponds to a reduction step of ${}^{d}\mathcal{X}$.

## 1.2 The main contributions

Here we will summarize the main contributions of this thesis' research.

**I** Classical erasure and duplication (${}^{*}\mathcal{X}$ calculus)

- A calculus for classical logic is presented with *name* as basic notion. It provides a correspondence "à la" Curry-Howard to sequent calculus where the structural rules *weakening* and *contraction* are made explicit. We consider only the *linear* terms, since non-linear terms can be encoded into linear ones.

– To the best of our knowledge, this is the first classical calculus which explicitly implements both *erasure* and *duplication* of terms.

**II** The diagrammatic computation (${}^{d}\mathcal{X}$ calculus)

– A diagrammatic calculus for classical logic is presented with a *port* as a basic notion. Both *static and dynamic* aspects are defined.

– Static aspects and relation with sequent proofs were studied by Robinson, in the work which presents the proof-nets for classical logic, but not the dynamics of those nets. We show (in the third part) that the reduction steps of diagrammatic calculus correspond to reduction steps performed on sequent proofs, modulo restructuring by permuting the independent rules.

**III** Equivalent programs in classical computation (${}^{\copyright}\mathcal{X}$ calculus)

– A one-dimensional calculus as counterpart of the diagrammatic calculus is presented. Reduction rules are defined modulo the congruence relation on terms.

– We explicitly define *which* terms should be identified in ${}^{*}\mathcal{X}$, in order to move towards the two-dimensional ${}^{d}\mathcal{X}$ calculus. This is done by introducing congruence rules.

– We use this result to go further. We show that the reduction steps in the diagrammatic framework correspond to ${}^{\copyright}\mathcal{X}$-reduction steps, modulo the congruence rules defined in the previous point.

In the case of all three calculi, the computation preserves the set of *free names/ports* (interface preservation). For the typed versions of these calculi we have the property of *type preservation*.

# Part I

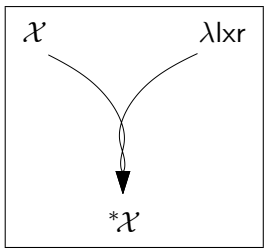
# Computing with Classical Sequents

# Chapter 2

# Introduction

The ${}^*\mathcal{X}$ calculus has been designed to provide a correspondence 'a là' Curry-Howard for the standard formulation of classical sequent calculus, with explicit structural rules (weakening and contraction). The direct correspondence between proofs and terms is achieved by using the technique of labeling formulas by names. These names are used to build terms so that the structure of a term captures the original structure of a corresponding proof. Furthermore, the computation of terms is defined in a way that mirrors the proof-transformation, that is, the cut-elimination.

The inspiration for ${}^*\mathcal{X}$ calculus comes from two sources. On the one hand, the direct predecessor is the classical term language called the $\mathcal{X}$ calculus. On the other hand, a very strong influence comes from the intuitionistic field and most notably the work on the $\lambda$lxr-calculus.




In our study we try to respect the underlying principles of these works, and implement them in a way that preserves their good properties too.

## About the $\mathcal{X}$ calculus

The $\mathcal{X}$ calculus is a relatively new term language, introduced in [vBLL05] and studied in more detail in [vBL07]. It is a low level language and it has

been shown that it can easily encode various other calculi. It captures the structure of classical proofs represented in the sequent calculus, as well as cut-elimination. Some of its properties are non-determinism, non-confluence and strong normalization (see [UB01a]) for typed terms.

Some closely related computational interpretations have been presented earlier. First of them is the so-called local cut-elimination procedure presented in [UB01a]. It is one of the three cut-elimination procedures studied in detail in [Urb00]. A term assignment is given for proofs in the classical sequent calculus (formulated with completely implicit structural rules). Then this term language was used as a tool to show the properties of classical sequent calculus. Most importantly, it enabled the authors to use the term-rewriting techniques in order to prove the strong normalization of cut-elimination in classical logic.

A second computational interpretation, very close to $\mathcal{X}$, has been presented by Lengrand in [Len03], under the name $\lambda\xi$-calculus. There it was studied in relation with $\bar{\lambda}\mu\tilde{\mu}$-calculus [CH00], and it was used to infer the strong normalization for $\bar{\lambda}\mu\tilde{\mu}$.

Although there are differences these three formulations are very close to each other. The syntaxes of $\lambda\xi$ and $\mathcal{X}$ are the same (there are minor differences such as the use of $\dagger_{\mathrm{V}}$ in the first, instead of $\not\leftarrow$ in the second). Both the syntax and the reduction rules of $\lambda\xi$ are said to be (in [Len03]) the subsystems of Urban's local cut-elimination procedure $(\mathcal{T}^{\leftrightarrow}, \xrightarrow{loc})$ (see [Urb00]). However, some differences at the level of reduction systems do exist. The so-called logical rules are identical in $\lambda\xi, \mathcal{X}$, and those which specify the logical cut with an axiom differ from Urban's approach (which we addopt in ${}^{*}\mathcal{X}$). Urban also includes the rule for garbage collection, which does not appear in neither $\lambda\xi$ nor $\mathcal{X}$. Then, there is also a rule for cut-propagation over a capsule (as an exception rule from general propagation philosophy) which appears in Urban's work and $\lambda\xi$, but this time it is not included in the $\mathcal{X}$ calculus. Finally there is a subtle difference at the level of activation rules as presented by Urban, namely he has an extra condition which does not allow activation towards a term which has an active cut at the top level.

We recall here the philosophy behind these calculi.

The study of Christian Urban [Urb00] is partially inspired by the work of Danos et al. presented in [DJS97] which considers Gentzen's sequent calculus as a programming language. The cut-elimination procedure is called $\mathrm{LK}^{tq}$ and it has properties of strong normalization, confluence and it is strongly connected to linear logic's proof nets. The confluence is obtained

by assigning color annotations to formulas, which restricts cut-reductions so that the critical pair does not arise. Confluence is essential in $\mathrm{LK}^{tq}$ because it enabled the authors to exploit the strong normalization result of proof nets in linear logic. However Urban's work is impressive as it reveals all the details of the complex classical cut-elimination, by developing a term-notation for proofs which is used as a tool, whereas in $\mathrm{LK}^{tq}$ many things are presented informally.

Moreover, it has been shown in [Urb00, UB01b], using the results of [BBS97], that not all normal forms are reachable using the $\mathrm{LK}^{tq}$ interpretation. Secondly, the restrictions introduced by using the colors are not needed to ensure strong normalization.

A departure was made from the traditional doctrine coming from intuitionistic logic, where computation is an equality preserving operation on proofs. Instead, the view was accepted that cut-elimination may or may not preserve the equality between proofs, and that the non-determinism is a natural property of classical logic. Urban's term language has been designed in accordance with the following principles:

- The cut-elimination procedure should not restrict the collection of normal forms (more precisely, essential normal forms) that are reachable.
- The cut-elimination procedure should be strongly-normalizing.
- The cuts should be allowed to pass over other cuts (that is, over other inactive cuts).

There are however a few restrictions. Namely, active cuts are not allowed to pass over other active cuts, and once the cut is activated it can not be deactivated at will (the propagation has to continue in the same direction).

## About the $\lambda$lxr-calculus

Although mainly concerned with computational aspects in the framework of classical logic, the inspiration for the results presented in this thesis came partially from the intuitionistic field. Primarily, it was the experience of studying $\lambda$lxr-calculus presented in [KL05, KL07] which had a significant influence.

The way $\lambda$lxr-calculus introduces terms for erasure and duplication into the calculus of explicit substitution - $\lambda$x [BR95] is similar to the way the ${}^{*}\mathcal{X}$ calculus introduces those terms with respect to the $\mathcal{X}$-calculus. According

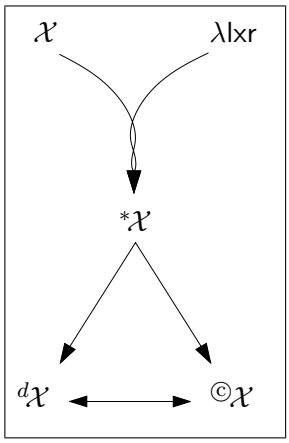
to what has been said one may notice that the intuitionistic calculi, $\lambda\mathsf{x}$ and $\lambda\mathsf{lxr}$, stand in the similar relation as the classical calculi $\mathcal{X}$ and ${}^{*}\mathcal{X}$. Moreover, if we consider the so-called propagation rules (which exist in $\mathcal{X}$ and ${}^{*}\mathcal{X}$) to be a classical analog to explicit substitution, then the calculi which do not have propagation rules, namely diagrammatic calculus (see Part II) and the ${}^{\copyright}\mathcal{X}$ (see Part III) could be related to $\lambda$-calculus which has an implicit substitution. This should not be taken strictly because the erasure and duplication are explicit in ${}^{d}\mathcal{X}$ and ${}^{d}\mathcal{X}$, which is not the case with the $\lambda$-calculus

The $\lambda\mathsf{lxr}$-calculus was created as an attempt to relate the two elementary decompositions. Namely, the decomposition of intuitionistic connectives in linear logic, and the decomposition of a meta-level $\lambda$-calculus substitution. The meta-substitution can be decomposed into more atomic steps, represented within the language [ACCL91], thus bringing the theoretical work closer to the actual implementations. It has been shown in [KL05] that there exists a very strong relation between $\lambda\mathsf{lxr}$-calculus and linear logic proof-nets.

The $\lambda\mathsf{lxr}$-calculus is a simple term language which extends the calculus of *explicit substitutions* $\lambda\mathsf{x}$ [BR95] with new operators for both *erasure* and *duplication*. The basic features of the $\lambda\mathsf{lxr}$-calculus are:

- Simple syntax which introduces explicit *eraser* and *duplicator*, and an intuitive operational semantics via equations and reduction rules.
- Sound and complete correspondence with a proof-net model. Moreover, reductions and equations have a natural correspondence with those of proof nets.
- Preservation of linearity and free variables. Preservation under reduction of assignable types. Step-by-step simulation of $\beta$-reduction. Preservation of strong normalization and strong normalization of typed terms. Confluence.

Many calculi implementing explicit substitutions were presented over the past years. It has been shown for some explicit versions of the $\lambda$-calculus that they did not preserve the good properties of its predecessors. For example, there are strongly normalizable terms in the $\lambda$-calculus, that are not strongly normalizable in $\lambda\sigma$ presented in [ACCL91]. Although it is a low level language, $\lambda\mathsf{lxr}$-calculus attempts to retain the good properties of its predecessors. The comparison of various approaches with respect to the calculi of explicit substitutions is elaborated by Kesner in [Kes07]. The $\lambda\mathsf{lxr}$-calculus is presented in more detail in Section 4.2 (page 28).




The $^{*}\mathcal{X}$ calculus is the core of this thesis research. From it, we will later derive the diagrammatic $^{d}\mathcal{X}$ calculus, and also its one-dimensional counterpart, the ©$\mathcal{X}$ calculus.

# Chapter 3

# From sequents to terms

The ${}^*\mathcal{X}$ calculus (presented in Chapter 5) has been designed to be in the closest correspondence with classical propositional logic represented in the sequent calculus formalism. This means that the ${}^*\mathcal{X}$ calculus captures both the structure and the dynamics of classical sequent proofs. Therefore, the first sections of this chapter are dedicated to the sequent calculus. We first recall the basic notions related to sequent calculus, as well as the main sequent systems for classical logic.

## 3.1 Classical sequent calculus

The *Sequent calculus* has been introduced by Gerhard Gentzen [Gen35] as a formalism more suitable for mathematical treatment than natural deduction. He also developed the *cut-elimination procedure* - a technique of proof transformation directed towards eliminating cut-rules. Gentzen introduced the sequent calculi for both intuitionistic and classical logic, denoted as LJ and LK, respectively. We will recall here some basic concepts of the sequent calculus for classical logic.

The central notion is this of *sequent*. A sequent in LK is a formal expression of the form $\Gamma \vdash \Delta$, where $\Gamma$ and $\Delta$ are called *contexts* and, technically, they can be treated either as lists, multisets or sets of formulas, depending on the particular formulation of the system. The sequents are usually written as

$$P_1, ..., P_m \vdash Q_1, ..., Q_n, \;\; m, n \geq 0$$

The zone on the left hand side of the turnstyle $\vdash$ is called *antecedent* and on the right-hand side the *succedent*. The formulas are of the first-order

predicate logic (one may also restrict this to propositional logic). We will focus on studying only the *implicative fragment* of the system.

Sequent calculus is supposed to consist of pure symbol manipulation, but there is, however, an intuition behind. The above sequent has an intuitive meaning that from the conjunction of the formulas in the antecedent, follows the disjunction of the formulas in the succedent, or even simpler "if all of $P_1, ..., P_m$ are true, at least one of $Q_1, ..., Q_n$ is true":

$$P_1 \wedge ... \wedge P_m \vdash Q_1 \vee ... \vee Q_n, \;\; m, n \geq 1$$

This interpretation extends to the case when $m = 0, n = 0$, by regarding the conjunction of zero formulas as *true*, and the disjunction of zero formulas as *false*.

*Inference rules* consist of sequents forming *premises* (above the horizontal line) and a *conclusion* (below the line). There can be zero, one, or two premises and there is always a single conclusion. There are three kinds of inference rules, namely *logical*, *structural* and *cut-rule*. We also distinguish *left* and *right* rules, depending on the side they act on in the concluding sequent. An example of a logical inference rule is arrow introduction to the right:

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta} \; (R\to)$$

It has a single premise and a single conclusion. It *introduces* the formula $A \to B$ on the right-hand side of the conclusion, and this formula is called the *principal formula* of that inference rule. Formulas that are shown explicitly in the premise, $A$ and $B$, are called *active formulas*[1] and they are involved in forming the principal formula. The other formulas in the contexts are called *side formulas*.

The *proofs* in the sequent calculus formalism are tree-structures[2], having inference rules as nodes, while the leaves are *axiom rules*, i.e., inference rules with no premises. There are three kind of inference rules: *logical* rules (axiom and left and right-introduction rules), *structural* rules (weakening, contraction and exchange rules) and a *cut*-rule.

Gentzen proved that a sequent calculus satisfies a *cut-elimination* property [Gen35], which states that any sequent provable using the cut-rule, can be proved without use of this rule. The cut-elimination theorem is the central result establishing the significance of the sequent calculus. In particular,

[1] Gentzen uses the notion "side formulas" instead, which rather suggests an element of the context.

[2] According to Gentzen [Gen35], they are in fact directed acyclic graphs.

we have that the cut-free proofs in the sequent calculus satisfy the *subformula property*: the premises of the inference rules contain only subformulas of their respective conclusions. As a consequence, the *consistency* of classical logic is shown by the following simple method: the empty sequent, which corresponds to true implies false, can not be inferred by applying any rule but cut-rule, and therefore is not derivable.

## 3.2 The variants of classical systems

It is possible to define many variants of Gentzen sequent systems. Although it might seem confusing, there is a method on how these variants are obtained, and in general no particular variant is to be preferred over the others. The basic Genzen systems for classical and intuitionistic logic denoted as $G1, G2$ and $G3$ are formalized in [Kle52] and later revisited in [TS96]. In brief, the essential difference between $G1$ and $G3$ is the presence or absence of explicit structural rules. The distinguishing point in the case of $G2$ is the use of the so-called mix instead of a cut rule. Although here we focus on the classical systems, we remark that the intuitionistic systems are obtained from classical ones by restricting sequents to having only one formula in the succedent.

Originally Gentzen presented a two-sided variant with explicitly given so-called structural rules, namely *weakening, contraction* and *exchange*. The exchange rule is explicit because the contexts are treated as lists.

$$\frac{\Gamma', A, B, \Gamma'' \vdash \Delta}{\Gamma', B, A, \Gamma'' \vdash \Delta}\ \textit{(L-exchange)} \qquad \frac{\Gamma \vdash \Delta', C, D, \Delta''}{\Gamma \vdash \Delta', D, C, \Delta''}\ \textit{(R-exchange)}$$

Structural rules can be hidden by reformulating the other rules in the system. We may define different representations by manipulating the presence of structural rules and the treatment of contexts. In general, if we wish to hide the exchange rule while keeping other structural rules, then the contexts should be treated as multisets. And if we decide to hide all structural rules, then it is reasonable to to treat contexts as sets.

In what follows we will revisit the $G$-systems in greater detail, taking into account only their implicative fragments.

**The system** $G1$ Among the three systems, $G1$ as presented by Kleene is the closest to the original formulation of Gentzen. Here we will keep the name $G1$, but the system will have exchange rules built-in the system[3] and

[3] In [Gen35, Kle52] exchange rules are presented explicitly.

contexts as multisets. This system is of importance for this thesis since, as it represents a logical setting for ${}^*\mathcal{X}$ calculus, the first calculi to be presented. The system $G1$ is given by Figure 3.1.

$$\frac{}{A \vdash A}\ (ax)$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', A \to B \vdash \Delta, \Delta'}\ (L\to) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\ (R\to)$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ (cut)$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ (weak\text{-}L) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ (weak\text{-}R)$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ (cont\text{-}L) \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ (cont\text{-}R)$$

Figure 3.1: Sequent system $G1$

To summarize, Figure 3.1 presents the classical system where symbols $A, B, ...$ are used to denote formulas and $\Gamma, \Delta, \Gamma', \Delta', ...$ are contexts, that are in this framework, multisets of formulas. Structural rules weakening and contraction are explicitly given. The axiom rule does not involve contexts. Inference rules with two premises are given in the context-splitting style. It has been shown [TS96] that we get the equivalent system in terms of provability if we apply context-sharing style.

**The system** $G2$ The system $G2$, as presented in [Kle52] is obtained from $G1$, the main difference being the replacement of cut rule with the *mix rule*. The mix is sometimes referred to as *multicut* and represents the generalization of a cut rule. The mix rule is given in the following way

$$\frac{\Gamma \vdash A^n, \Delta \qquad \Gamma', A^m \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ (mix)$$

The systems obtained in this way are equivalent as proven in [TS96] by showing that the rules mix and cut are interderivable in the presence of the explicit structural rules.

**The system** $G3$ The sequent system $G3$ is obtained from $G1$ by absorbing all structural rules into the remaining rules, by giving the remaining rules the appropriate form. In other words, there is no explicit structural rules. Structural rules are hidden in the form of the logical rules and cut-rule, and thus performed automatically.

This system has been mentioned as $G3a$ in [Kle52] and formalized as classical $G3$ in [TS96]. It is presented by Figure 3.2, where $A, B, ...$ range over formulas, while contexts $\Gamma, \Delta, ...$ are finite *sets* of formulas.

$$\frac{}{\Gamma, A \vdash A, \Delta}\ (ax)$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\ (L\to) \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \to B, \Delta}\ (R\to)$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ (cut)$$

Figure 3.2: Sequent system $G3$

Inference rules with two premises are given in the context-sharing style. The definition of the axiom rule involves contexts, thus allowing arbitrary formulas to be introduced at that level, i.e., weakening rule is hidden in the form of the axiom.

**System** $G1$ **with names** We will introduce the non-standard variant of a sequent system $G1$, having as a principal difference the use of pairs (name, formula) to build contexts, instead of pure formulas. A similar variant was presented in [UB01a, Urb00], only there it was inspired by the $G3$ system.

The idea is to label formulas, that is, to assign names to formulas and then use those names to form the terms of the calculus. If the formula is in the antecedent of a sequent then it is labeled by an inname $(x, y, z, ...)$ and otherwise by an outname $(\alpha, \beta, \gamma, ...)$. Innames and outnames in general will be called simply names[4]. So the contexts on the left-hand side of the turnstile are built from (inname, formula) pairs and on the right-hand style by (outname, formula) pairs. For example, contexts are

$$\Gamma = \{(x, A), (y, B), (z, C)\} \quad \text{and}$$
$$\Delta = \{(\alpha, E), (\beta : F)\}$$

[4]More about the notions of name, inname and outname will be said in Chapter 5

We say that $\{x, y, z\}$ is the *domain* of the context $\Gamma$, denoted $dom(\Gamma)$, and $\{\alpha, \beta\}$ is the domain of $\Delta$. In order to simplify the notation for contexts, instead of $\{(x, A), (y, B), (z, C)\}$ we will write simply $x : A, y : B, z : C$. The inference rules for this variant of the sequent calculus are given in Figure 3.3.

$$\frac{}{x : A \vdash \alpha : A}\ \textit{(axiom)}$$

$$\frac{\Gamma \vdash \alpha : A, \Delta \qquad \Gamma', y : B \vdash \Delta'}{\Gamma, \Gamma', x : A \to B \vdash \Delta, \Delta'}\ (L\to) \qquad \frac{\Gamma, x : A \vdash \alpha : B, \Delta}{\Gamma \vdash \beta : A \to B, \Delta}\ (R\to)$$

$$\frac{\Gamma \vdash \alpha : A, \Delta \qquad \Gamma', x : A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \textit{(cut)}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, x : A \vdash \Delta}\ \textit{(L-weakening)} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \alpha : A, \Delta}\ \textit{(R-weakening)}$$

$$\frac{\Gamma, x : A, y : A \vdash \Delta}{\Gamma, z : A \vdash \Delta}\ \textit{(L-contraction)} \qquad \frac{\Gamma \vdash \alpha : A, \beta : A, \Delta}{\Gamma \vdash \gamma : A, \Delta}\ \textit{(R-contraction)}$$

Figure 3.3: The system $G1$ with pairs

The system presented by Figure 3.3 needs to be explained in more detail. One of the most important things is how the formulas are labeled. To start, we do not want to allow the situation where one name is used to label two distinctive formulas. For example context $x : A$, $x : B$ is ill-formed. This restriction is referred to as the *context convention*.

In this framework we introduce the *naming convention*, which simply means that we label every occurrence of a formula with a different, fresh name. Recall that we start from the system $G1$, where contexts are multisets and one formula may occur several times in a context. Thus by assigning names to formulas in that way we get the system contexts are sets, that is, every pair (name, formula) occurs at most once.

We also have a *convention on commas*. The commas in both, premise and conclusion, are used to denote the disjoint set union. For example in the rule

$$\frac{\Gamma, x : A \vdash \alpha : B, \Delta}{\Gamma \vdash \beta : A \to B, \Delta}\ (R\to)$$

the comma in the conclusion is disjoint set union since $\beta$ is chosen to be a fresh name, and therefore there is no implicit contraction. In the premise, commas denote the disjoint set union, since it is assumed that there is no repetition of elements.

Two-premise inference rules in our calculus are given in the context-splitting style. Take as an example the rule

$$\frac{\Gamma \vdash \alpha : A, \Delta \qquad \Gamma', y : B \vdash \Delta'}{\Gamma, \Gamma', x : A \to B \vdash \Delta, \Delta'}\ (L\to)$$

When joining contexts we have to be careful not to break the context-convention, which would be the case when for example $z : C$, $z : D$ occur in the same context, and this can always be achieved by appropriate renaming. Another case to think about when joining the contexts, say $\Gamma$, and $\Gamma'$, is the possibility to get two identical pairs, say $z : C$, $z : C$, in the conclusion. We avoid this by appropriate renaming before making the disjoint set union $\Gamma, \Gamma'$.

## From sequent proofs to terms

Names that have been used to label formulas are employed to build terms of the ${}^*\mathcal{X}$ calculus. These terms capture the structure of proofs given in the sequent calculus as formulated previously. Moreover the computation in ${}^*\mathcal{X}$ calculus corresponds to proof-transformation (cut-elimination) in the sequent calculus (see Chapter 5).

# Chapter 4

# Related computational interpretations

## 4.1 The $\mathcal{X}$ calculus

This section presents the $\mathcal{X}$ calculus which is, together with the $\lambda$lxr-calculus, a predecessor of ${}^{*}\mathcal{X}$. The design of ${}^{*}\mathcal{X}$ calculus has been directly inspired by the $\mathcal{X}$ calculus, while the other two calculi presented in the thesis (called ${}^{d}\mathcal{X}$ and ${}^{\copyright}\mathcal{X}$) are indirectly influenced by $\mathcal{X}$ through the ${}^{*}\mathcal{X}$ calculus.

The $\mathcal{X}$ calculus is presented is van Bakel, Lescanne and Lengrand in [vBLL05]. The origin of the language is in the notations for classical sequent proofs by Urban [Urb00], introduced as a tool to express the cut-elimination procedure as a term rewriting system, which later allowed him to prove strong normalization of cut-elimination. A close variant of the language has been studied by Lengrand in relation with the $\bar{\lambda}\mu\tilde{\mu}$-calculus, via the calculus there called $\lambda\xi$ [Len03].

It is argued in [Urb00] that non-determinism, although it leads to non-confluence, should be considered as an intrinsic property of classical logic. This point of view was taken in some earlier works, for example [BB94, Her95, BBS97] and more recently in [Gir01, Hyl02]. This means that, in classical logic, we depart from the traditional intuitionistic (and linear) logic doctrine, where cut-elimination is an equality preserving operation on proofs.

### 4.1.1 The syntax

The $\mathcal{X}$ calculus corresponds to a sequent system with implicit structural rules (Figure 3.2, page 19). Since we consider only the implicative fragment, the only inference rules are axiom, cut, left-arrow introduction and right-arrow introduction. Therefore, in the $\mathcal{X}$ calculus there are four constructors. The syntax is presented by Figure 4.1.

| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $\langle x.\alpha \rangle$ | *capsule* |
| | $\mid$ | $\widehat{x}\, P\, \widehat{\beta} \cdot \alpha$ | *exporter* |
| | $\mid$ | $P\, \widehat{\alpha}\, [x]\, \widehat{y}\, Q$ | *importer* |
| | $\mid$ | $P \widehat{\alpha} \dagger \widehat{x} Q$ | *cut* |

Figure 4.1: The syntax of $\mathcal{X}$

The term *capsule* corresponds to an axiom rule, *cut* corresponds to a cut-rule, *importer* corresponds to left-arrow introduction rule and *exporter* corresponds to right-arrow introduction rule.[1] The syntax is then extended by two *active cuts* that reflect the non-deterministic *choice* which exists in the sequent calculus.

| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $\cdots$ | |
| | $\mid$ | $P \widehat{\alpha} \nearrow \widehat{x} Q$ | *left-active cut* |
| | $\mid$ | $P \widehat{\alpha} \nwarrow \widehat{x} Q$ | *right-active cut* |

In Urban's work where term annotation for proofs are defined, his *raw* terms correspond to $\mathcal{X}$-terms in the following way:

$$
\begin{array}{lcl}
\langle x.\alpha \rangle & = & \mathrm{Ax}(x, \alpha) \\
\widehat{x}\, P\, \widehat{\beta} \cdot \alpha & = & \mathrm{Imp}_R((x)\langle \beta \rangle M, \alpha) \\
P\, \widehat{\alpha}\, [x]\, \widehat{y}\, Q & = & \mathrm{Imp}_L(\langle \alpha \rangle P, (y)Q, x) \\
P \widehat{\alpha} \dagger \widehat{x} Q & = & \mathrm{Cut}(\langle \alpha \rangle P, (x)Q)
\end{array}
$$

He latter adds *labeled cuts*: $\overleftarrow{\mathrm{Cut}}(\langle \alpha \rangle P, (x)Q)$ to denote left-active version and $\overrightarrow{\mathrm{Cut}}(\langle \alpha \rangle P, (x)Q)$ to denote right-active version of the $\mathrm{Cut}(\langle \alpha \rangle P, (x)Q)$.

### 4.1.2 The computation

The reduction procedure defined on $\mathcal{X}$-terms corresponds to the process of cut-elimination in the sequent calculus. There are 20 reduction rules, which

[1] In the original papers importer and exporter were called import and mediator.

are split into logical, activation and propagation groups. The way the rules are grouped differs slightly from the approach in [vBLL05, vBL07]. Also, the naming of rules is adjusted so as to ease the comparison with ${}^*\mathcal{X}$-rules.

### Logical rules

Logical rules define how to eliminate a cut, when both names cut is referring to are *freshly introduced*.

**Definition 1 (Fresh introduction)** *The notion of* fresh introduction *is defined as follows:*

*– The term $P$ freshly introduces $x$ if $P = \langle x.\alpha \rangle$ or $P = Q\,\widehat{\alpha}\,[x]\,\widehat{y}\,R$, with $x \notin N(Q), x \notin N(R)$.*
*– The term $P$ freshly introduces $\alpha$ if $P = \langle x.\alpha \rangle$ or $P = \widehat{x}\,Q\,\widehat{\beta} \cdot \alpha$, with $\alpha \notin N(Q)$.*

Informally, names are freshly introducesd if they appear once and only once, at the top level of their corresponding terms.[2] The cut in this position can not be activated. Logical rules are shown by Figure 4.2.

$$
\begin{array}{lcll}
(\textit{cap-ren}) & : & \langle y.\alpha \rangle \widehat{\alpha} \dagger \widehat{x}\langle x.\beta \rangle & \rightarrow \quad \langle y.\beta \rangle \\
(\textit{exp-ren}) & : & (\widehat{y}\,P\,\widehat{\beta} \cdot \alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma \rangle & \rightarrow \quad \widehat{y}\,P\,\widehat{\beta} \cdot \gamma, \qquad \alpha \notin N(P) \\
(\textit{imp-ren}) & : & \langle y.\alpha \rangle \widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) & \rightarrow \quad Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R, \;\; x \notin N(Q), \; x \notin N(R) \\
(\textit{ei-insert}) & : & (\widehat{y}\,P\,\widehat{\beta} \cdot \alpha)\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R) & \rightarrow \quad \textit{either} \left\{ \begin{array}{l} (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \\ Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \end{array} \right. \\
 & & \alpha \notin N(P), \; x \notin N(Q), \; x \notin N(R) &
\end{array}
$$

Figure 4.2: Logical rules in $\mathcal{X}$

The first three rules specify a *renaming*. The last rule is called *insertion* and defines an interaction between an importer and an exporter. It inserts an immediate subterm of an exporter between two immediate subterms of an importer.

[2] This is more complex then in ${}^*\mathcal{X}$, where the linearity condition guarantees that if a name occurs at the top level, then it does not occur elsewhere.

### Activation rules

The activation rules describe the non-deterministic choice which exists in classical cut-elimination. If a cut refers to names which both do not occur in their corresponding terms, then the choice is also a source of non-confluence. This choice has usually been bypassed in the earlier interpretations, either by restricting the reduction procedure (a very common one is not to allow cuts to pass over other active cuts), or by giving priority to a certain strategy (as in [DJS96], by assigning colors to formulas).

Here, the cut can be activated in both directions only if both conditions are fulfilled at the same time, as shown by Figure 4.3.

$$\begin{array}{lcl} (act\text{-}L) & : & P\widehat{\alpha} \dagger \widehat{x}Q \ \rightarrow P\widehat{\alpha} \nearrow \widehat{x}Q, \ \text{if } \alpha \text{ not freshly introduced by } P \\ (act\text{-}R) & : & P\widehat{\alpha} \dagger \widehat{x}Q \ \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}Q, \ \text{if } x \text{ not freshly introduced by } Q \end{array}$$

Figure 4.3: Activation rules in $\mathcal{X}$

**Remark** The activation is followed by some of the propagation rules (see Figures 4.4 and 4.5).

### Propagation rules

*Left* and *right propagation* rules are given in Figures 4.4 and 4.5, respectively. Numerous propagation rules describe how a cut is being pushed through the structure of terms, but also define situations when deactivation, erasure and duplication occurs. This means that in the $\mathcal{X}$ calculus several actions can be defined by a single reduction rule. Take for example the rule from Figure 4.5 which involves propagation, duplication and deactivation:

($\nwarrow$-*prop-dupl-deact*) :

$$P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) \rightarrow P\widehat{\alpha} \dagger \widehat{x}((P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\,[x]\,\widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R))$$

In the ${}^*\mathcal{X}$ calculus these operations are strictly separated, in a sense that one reduction rule only describes one action. The rule above is split into three cases, due to linearity which allows at most one occurrence of a free name in a term. Therefore, instead of the above rule, in ${}^*\mathcal{X}$ we have:

$$
\begin{array}{lcl}
(\nwarrow\textit{-deact}) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) \;\rightarrow\; P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) \\
(\nwarrow\textit{imp-prop}_1) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\rightarrow\; (P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,R,\;\; x \neq y, x \in N(Q) \\
(\nwarrow\textit{imp-prop}_2) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\rightarrow\; Q\,\widehat{\beta}\,[y]\,\widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R),\;\; x \neq y, x \in N(R)
\end{array}
$$

Left propagation rules define computation when a cut is left-activated. There is a special rule to describe propagation over a capsule (whose both names are bound by cuts), labeled ($\nearrow$-($c$)-*prop-deact*). This rule is introduced to prevent possible infinite reductions, for which the reasons are of syntactic nature.

$$
\begin{array}{lcrcl}
(\nearrow\textit{-eras}) & : & \langle x.\alpha\rangle\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & \langle x.\alpha\rangle, \quad \alpha \neq \beta \\
(\nearrow\textit{-deact}) & : & \langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & \langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R \\
(\nearrow\textit{-prop}) & : & (\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & \widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha, \quad \alpha \neq \beta \\
(\nearrow\textit{-prop-dupl-deact}) & : & (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & (\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R \\
(\nearrow\textit{-prop-dupl}_1) & : & (P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & (P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\alpha}\,[x]\,\widehat{z}\,(Q\widehat{\beta} \nearrow \widehat{y}R) \\
(\nearrow\textit{-}(c)\textit{-prop-deact}) & : & (P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{y}R \\
(\nearrow\textit{-prop-dupl}_2) & : & (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow & (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \nearrow \widehat{y}R), \quad Q \neq \langle x.\beta\rangle
\end{array}
$$

Figure 4.4: Left propagation (erasure/duplication/deactivation) in $\mathcal{X}$

$$
\begin{array}{lcll}
(\nwarrow\textit{-eras}) & : & P\widehat{\alpha} \nwarrow \widehat{x}\langle y.\beta\rangle & \rightarrow \langle y.\beta\rangle, \quad x \neq y \\
(\nwarrow\textit{-deact}) & : & P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle & \rightarrow P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \\
(\nwarrow\textit{-prop}) & : & P\widehat{\alpha} \nwarrow \widehat{x}(\widehat{y}\,Q\,\widehat{\gamma}\cdot\beta) & \rightarrow \widehat{y}\,(P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\gamma}\cdot\beta \\
(\nwarrow\textit{-prop-dupl}_1) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) & \rightarrow (P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad x \neq y \\
(\nwarrow\textit{-prop-dupl-deact}) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) & \rightarrow P\widehat{\alpha} \dagger \widehat{x}((P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\,[x]\,\widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R)) \\
(\nwarrow\textit{-}(c)\textit{-prop-deact}) & : & P\widehat{\alpha} \nwarrow \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R) & \rightarrow P\widehat{\alpha} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R) \\
(\nwarrow\textit{-prop-dupl}_2) & : & P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \rightarrow (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad Q \neq \langle x.\beta\rangle
\end{array}
$$

Figure 4.5: Right propagation (erasure/duplication/deactivation) in $\mathcal{X}$

### 4.1.3 The type system

The type assignment system for the $\mathcal{X}$ calculus is given by Figure 4.6.

$$\frac{}{\langle x.\alpha\rangle :\cdot\ \Gamma, x : A \vdash \alpha : A, \Delta}\ (axiom)$$

$$\frac{P :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad Q :\cdot\ \Gamma, x : B \vdash \Delta}{P\,\widehat{\alpha}\ [y]\ \widehat{x}\,Q :\cdot\ \Gamma, y : A \to B \vdash \Delta}\ (\to L) \qquad \frac{P :\cdot\ \Gamma, x : A \vdash \alpha : B, \Delta}{\widehat{x}\,P\,\widehat{\alpha}\cdot\beta :\cdot\ \Gamma \vdash \beta : A \to B, \Delta}\ (\to R)$$

$$\frac{P :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad Q :\cdot\ \Gamma, x : A \vdash \Delta}{P\widehat{\alpha}\ \dagger\ \widehat{x}Q :\cdot\ \Gamma \vdash \Delta}\ (cut)$$

Figure 4.6: The type system for $\mathcal{X}$

## 4.2 The λlxr-calculus

In this section we present the intuitionistic λlxr-calculus [KL06, KL07], with its syntax, congruence relation and reduction relation. Besides $\mathcal{X}$, this calculus was another starting point for the design of ${}^{*}\mathcal{X}$ calculus. The approach which was taken in λlxr when introducing terms for erasure and duplication is similar to the the one taken when adding those terms to $\mathcal{X}$.

The λlxr-calculus was created as an attempt to relate the two elementary decompositions. On the one hand, there exists a decomposition of intuitionistic connectives in linear logic [Gir87]. On the other hand, the principal reduction rule of λ-calculus [Chu85], can be decomposed into more elementary steps. This is due to a fact that β-reduction includes a meta-level substitution, which, by assigning it a syntactic representation, can be manipulated explicitly [ACCL91], thus breaking β-reduction into more atomic steps. This decomposition of a meta-level substitution brings the theoretical work closer to practical implementations. The deep connection between these decompositions is shown by relating this calculus to linear logic *proof-nets* [Gir87].

The λlxr-calculus is a simple term language which extends the calculus of *explicit substitutions* λx [BR95], with new operators for both *erasure* and *duplication*. This calculus has been designed to provide a sound and complete correspondence with the intuitionistic fragment of linear logic proof-nets. The basic features of λlxr-calculus are:

- Simple syntax which introduces explicit *eraser* and *duplicator*, and intuitive operational semantics via equations and reduction rules.
- Sound and complete correspondence with a proof-net model. Moreover, reductions and equations have a natural correspondence with

those of proof nets.

- Preservation of linearity and free variables, and for typed language preservation of types.
- Full composition of explicit substitutions.
- Preservation of strong normalization.
- Strong normalization of typed terms.
- Step-by-step simulation of $\beta$-reduction.
- Confluence.

Many calculi implementing explicit substitutions were presented over the past years. The result of Melliès [Mel95] has shown that there are strongly normalizable terms in the $\lambda$-calculus that are not strongly normalizable in some explicit versions of the $\lambda$-calculus; for example see [ACCL91]. Motivated to avoid this phenomenon, various approaches were taken to impose restrictions, which usually lead to losing some of the desirable properties. The ideas presented in the work of David and Guillaume on $\lambda_{ws}$-calculus [DG01], were of particular importance for λlxr-calculus, which tries to implement its underlying language without loosing its good properties. Kesner [Kes07] presents a very simple calculus of explicit substitutions which preserves strong normalization and faithfully implements implicit substitution.

### 4.2.1 The syntax

The *syntax* for raw terms is given by the grammar presented in Figure 4.7.

| | | | |
|---|---|---|---|
| $t$ | $::=$ | $x$ | *variable* |
| | $\mid$ | $\lambda x.t$ | *abstraction* |
| | $\mid$ | $t\,t$ | *application* |
| | $\mid$ | $t\langle x = t\rangle$ | *closure* |
| | $\mid$ | $\mathrm{W}_x(t)$ | *weakening* |
| | $\mid$ | $\mathrm{C}_x^{y,z}(t)$ | *contraction* |

Figure 4.7: The syntax of the λlxr-calculus

A set of *variables* $x, y, z, ...$ is assumed to be in bijection with natural numbers and thus equipped with a *total order*.

The variable $x$ is bound in the terms $\lambda x.t$ and $t\langle x = t\rangle$, whilst $\mathrm{C}_x^{y,z}(t)$ binds the variables $y$ and $z$ in $t$. In the terms $\mathrm{W}_x(t)$ and $\mathrm{C}_x^{y,z}(t)$, the variable $x$ is free. $fv(t)$ is used to denote a set of free variables of $t$.

**The notion of linearity** The term is said to be *linear* if it satisfies the following two conditions:

- in every subterm, every variable has at most one free occurrence, and
- every binder does bind a free occurrence of a variable (and therefore only one, if we take into consideration the first point).

In general, every non-linear term produced by the syntax can be translated into linear one. This has been shown in Section 4 of [KL07], which studies the relation between $\lambda$-calculus and $\lambda\mathsf{lxr}$-calculus.

### 4.2.2 Congruence equations for $\lambda\mathsf{lxr}$-terms

A *congruence relation* is defined on terms, denoted $\equiv$. It is the smallest reflexive, symmetric, transitive and context-closed relation induced by the rules given in Figure 4.8.

| | | | |
|---|---|---|---|
| $\mathrm{C}_w^{x,v}(\mathrm{C}_x^{y,z}(t))$ | $\equiv_{\mathsf{A_C}}$ | $\mathrm{C}_w^{x,y}(\mathrm{C}_x^{z,v}(t))$ | |
| $\mathrm{C}_x^{y,z}(t)$ | $\equiv_{\mathsf{C_c}}$ | $\mathrm{C}_x^{z,y}(t)$ | |
| $\mathrm{C}_{x'}^{y',z'}(\mathrm{C}_x^{y,z}(t))$ | $\equiv_{\mathsf{P_C}}$ | $\mathrm{C}_x^{y,z}(\mathrm{C}_{x'}^{y',z'}(t))$ | if $x \neq y', z'$ and $x' \neq y, z$ |
| $\mathrm{W}_x(\mathrm{W}_y(t))$ | $\equiv_{\mathsf{P_w}}$ | $\mathrm{W}_y(\mathrm{W}_x(t))$ | |
| $t\langle x = u\rangle\langle y = v\rangle$ | $\equiv_{\mathsf{P_s}}$ | $t\langle y = v\rangle\langle x = u\rangle$ | if $y \notin fv(u)$ and $x \notin fv(v)$ |
| $\mathrm{C}_w^{y,z}(t)\langle x = u\rangle$ | $\equiv_{\mathsf{P_{cs}}}$ | $\mathrm{C}_w^{y,z}(t\langle x = u\rangle)$ | if $x \neq w$ and $y, z \notin fv(t)$ |

Figure 4.8: Congruence axioms for $\lambda\mathsf{lxr}$-terms

The rule $\mathsf{A_c}$ describes the internal *associativity* of contraction. Similarly, the rule $\mathsf{C_c}$ expresses the internal *commutativity* of contraction. Weakenings, contractions and substitutions can also be commuted on their own; this is referred to as *permutability*. The equations $\mathsf{P_c}$, $\mathsf{P_w}$ and $\mathsf{P_s}$ express the permutability of independent contractions, weakenings, and substitutions, respectively. The equation $\mathsf{P_{cs}}$ expresses the permutability between independent contraction and substitution. Notice that the equation $\mathsf{A_c}$ needs the side condition $x \neq y, v$, while $\mathsf{P_{cs}}$ needs $x \neq y$, but these cases are in the scope of Barendregt's convention which is assumed.

**General properties of ≡** The congruence relation ≡ satisfies the following properties. It preserves the set of free variables and linearity. It also preserves types if we speak about typed calculus (see Figure 4.9). Besides that, each congruence rule induces a congruence class, and each congruence class contains finitely many terms. For more details see Sections 2.3 and 2.5 in [KL07].

### 4.2.3 The reduction relation

The *reduction relation* of the λlxr-calculus is the relation generated by the reduction rules presented on the next page, modulo the congruence relation presented in Figure 4.8.

**The notation** The notation $\Phi, \Upsilon, \Sigma, \Psi...$ is used to denote finite *lists* of variables (each of the symbols denotes a list), with no repetition. There exists a *renaming operation* denoted as $\mathcal{R}^{\Phi}_{\Psi}(t)$, which is defined as renaming of $\Psi$ by $\Phi$ on a term $t$. This means, if we assume that $\Phi = (x_1, ..., x_n)$, $\Psi = (y_1, ..., y_n)$, that we perform simultaneous substitutions of every free occurrence of $y_i$ by $x_i$, for $i = 1, ..., n$. In the case of ${}^*\mathcal{X}$, a renaming operation is used in the same situations, but it is implemented by indexing names in order to simplify the notation.

The notation $\mathrm{C}^{\Psi,\Upsilon}_{\Phi}(t)$ and $\mathrm{W}_{\Psi}(t)$ is introduced to generalize contraction and weakening from variables to lists of variables. This is done for simplicity reasons. In the case of empty lists we have $\mathrm{C}^{\emptyset,\emptyset}_{\emptyset}(t) = t$ and $\mathrm{W}_{\emptyset}(t) = t$.

Sometimes a set of variables is used in places where a list is expected in order to make formulation of the primarily reduction rules more understandable. There is no ambiguity, as a set of variables can always be viewed as a list, according to the total order that exists on the set of variables.

**General properties of reduction relation** Similarly to congruence rules, the reduction rules satisfy the preservation properties: the reduction rules preserve the *set of free variables* and *linearity*. Then there are properties of confluence, step-by-step preservation of $\beta$-reduction and, for the typed language (see Figure 4.9), preservation of strong normalization, strong normalization and subject reduction property.

| | | | | |
|---|---|---|---|---|
| $B$ | $(\lambda x.t)u$ | $\longrightarrow$ | $t\langle x = u\rangle$ | |
| System x | | | | |
| Abs | $(\lambda y.t)\langle x = u\rangle$ | $\longrightarrow$ | $\lambda y.(t\langle x = u\rangle)$ | |
| $\mathsf{App}_1$ | $(tv)\langle x = u\rangle$ | $\longrightarrow$ | $t\langle x = u\rangle v$ | $x \in fv(t)$ |
| $\mathsf{App}_2$ | $(tv)\langle x = u\rangle$ | $\longrightarrow$ | $tv\langle x = u\rangle$ | $x \in fv(u)$ |
| Var | $x\langle x = u\rangle$ | $\longrightarrow$ | $u$ | |
| $\mathsf{Weak}_1$ | $\mathrm{W}_x(t)\langle x = u\rangle$ | $\longrightarrow$ | $\mathrm{W}_{fv(u)}(t)$ | |
| $\mathsf{Weak}_2$ | $\mathrm{W}_y(t)\langle x = u\rangle$ | $\longrightarrow$ | $\mathrm{W}_y(t\langle x = u\rangle)$ | $x \neq y$ |
| Cont | $\mathrm{C}_x^{y,z}(t)\langle x = u\rangle$ | $\longrightarrow$ | $\mathrm{C}_{fv(u)}^{\Psi,\Upsilon}(t\langle y = u_1\rangle\langle z = u_2\rangle)$ | |
| | | where $\Psi, \Upsilon$ are fresh, | $u_1 = \mathcal{R}_\Psi^{fv(u)}$, $u_2 = \mathcal{R}_\Upsilon^{fv(u)}$ | |
| Comp | $t\langle y = v\rangle\langle x = u\rangle$ | $\longrightarrow$ | $t\langle y = v\langle x = u\rangle\rangle$ | $x \in fv(v)$ |
| System r | | | | |
| WAbs | $\lambda x.\mathrm{W}_y(t)$ | $\longrightarrow$ | $\mathrm{W}_y(\lambda x.t)$ | $x \neq y$ |
| WApp1 | $\mathrm{W}_y(u)v$ | $\longrightarrow$ | $\mathrm{W}_y(uv)$ | |
| WApp2 | $u\mathrm{W}_y(v)$ | $\longrightarrow$ | $\mathrm{W}_y(uv)$ | |
| WSubs | $t\langle x = \mathrm{W}_y(u)\rangle$ | $\longrightarrow$ | $\mathrm{W}_y(t\langle x = u\rangle)$ | |
| Merge | $\mathrm{C}_w^{y,z}(\mathrm{W}_y(t))$ | $\longrightarrow$ | $\mathcal{R}_w^z(t)$ | |
| Cross | $\mathrm{C}_w^{y,z}(\mathrm{W}_x(t))$ | $\longrightarrow$ | $\mathrm{W}_x(\mathrm{C}_w^{y,z}(t))$ | $x \neq y,\ x \neq z$ |
| CAbs | $\mathrm{C}_w^{y,z}(\lambda x.t)$ | $\longrightarrow$ | $\lambda x.\mathrm{C}_w^{y,z}(t)$ | |
| CApp1 | $\mathrm{C}_w^{y,z}(tu)$ | $\longrightarrow$ | $\mathrm{C}_w^{y,z}(t)\, u$ | $y, z \in fv(t)$ |
| CApp2 | $\mathrm{C}_w^{y,z}(tu)$ | $\longrightarrow$ | $t\, \mathrm{C}_w^{y,z}(u)$ | $y, z \in fv(u)$ |
| CSubs | $\mathrm{C}_w^{y,z}(t\langle x = u\rangle)$ | $\longrightarrow$ | $t\langle x = \mathrm{C}_w^{y,z}(u)\rangle$ | $y, z \in fv(u)$ |

### 4.2.4 Typing rules

This section presents the simply-typed $\lambda$-calculus. *Types* are defined by the following grammar, where $\sigma$ ranges over a countable set of atomic types.

$$A ::= \sigma \ \mid A \to A$$

An environment $\Gamma$ is a finite mapping from variables to types, i.e., a finite set of pairs $x : A$. Comma in the expression $\Gamma, \Delta$ denotes the disjoint union of the environments $\Gamma$ and $\Delta$. A *judgement* is an object of the form $\Gamma \vdash t : A$, where $\Gamma$ is an environment, $t$ is a $\lambda$lxr-term and $A$ is a type.

The *typing rules* of the simply-typed $\lambda$lxr-calculus are shown in Figure 4.9.

$$\frac{}{x : A \vdash x : A}\ (axiom) \qquad \frac{\Gamma, x : B \vdash t : A \qquad \Delta \vdash M : B}{\Gamma, \Delta \vdash t\langle x = M\rangle : A}\ (subs)$$

$$\frac{\Gamma \vdash t : A \to B \quad \Delta \vdash v : A}{\Gamma, \Delta \vdash (tv) : B}\ (app) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (lambda)$$

$$\frac{\Gamma, x : A, y : A \vdash M : B}{\Gamma, z : A \vdash \mathrm{C}_z^{x,y}(M) : B}\ (cont) \qquad \frac{\Gamma \vdash t : A}{\Gamma, x : B \vdash \mathrm{W}_x(t) : A}\ (weak)$$

Figure 4.9: Typing rules for the $\lambda$lxr-terms

## 4.3 The $\bar{\lambda}\mu\tilde{\mu}$-calculus

The $\bar{\lambda}\mu\tilde{\mu}$-calculus is introduced by Curien and Herbelin [CH00] as one of the first calculi to provide a Curry-Howard correspondence with the Gentzen's sequent calculus for classical logic. It can be seen as a symmetric variant of Parigot's $\lambda\mu$-calculus [Par92], which is a natural deduction calculus, extended to classical framework by enabling to deal with multiple conclusions.

The *symmetries* are expected to exist in calculi related to sequent calculus, and in $\bar{\lambda}\mu\tilde{\mu}$ they are of two kinds: *term/evaluation* context and *call-by-name/call-by-value*. They can be seen as dualities, the former being the static, and the later dynamic duality. Computational symmetries were exhibited in some earlier works [BB94, Fil89].

There exists a version of $\bar{\lambda}\mu\tilde{\mu}$-calculus *with explicit substitution*, introduced first in [Her01] and later studied in the context of strong normalization in [Pol04].

**The syntax** The basic constructs are called *commands, terms*, and *contexts*. The syntax of $\bar{\lambda}\mu\tilde{\mu}$ is given by Figure 4.10, where $c$ ranges over the set of commands, $v$ ranges over the set of terms and $e$ ranges over the set of contexts:

$$\begin{array}{lcl} c & ::= & \langle v \parallel e \rangle \\ v & ::= & x \mid \lambda x.v \mid \mu\alpha.c \\ e & ::= & \alpha \mid v \cdot e \mid \tilde{\mu}x.c \end{array}$$

Figure 4.10: The syntax of $\bar{\lambda}\mu\tilde{\mu}$-calculus

There are two kinds of variables: *term variables*, denoted by $x, y, \ldots$ and *context variables*, denoted by $\alpha, \beta, \ldots$. Term variables can be bound by $\lambda$-abstraction or by $\tilde{\mu}$-abstraction, whereas the context variable can be bound by $\mu$-abstraction ($\lambda, \mu$ and $\tilde{\mu}$ are binders). The sets of free term and context variables are defined as usual, respecting Barendregt's convention [Bar84] that no variable can be both bound and free in the term.

**Reduction rules** There are only three rules that characterize the reduction in $\bar{\lambda}\mu\tilde{\mu}$: The above substitutions are defined as to avoid variable cap-

$$\begin{array}{lrcl} (\rightarrow') & \langle \lambda x.v_1 \parallel v_2 \cdot e \rangle & \rightarrow & \langle v_2 \parallel \tilde{\mu}x.\langle v_1 \parallel e \rangle \rangle \\ (\mu) & \langle \mu\alpha.c \parallel e \rangle & \rightarrow & c[e/\alpha] \\ (\tilde{\mu}) & \langle v \parallel \tilde{\mu}x.c \rangle & \rightarrow & c[v/x] \end{array}$$

Figure 4.11: The reduction rules of $\bar{\lambda}\mu\tilde{\mu}$-calculus

ture [Bar84].

The calculus has a critical pair $\langle \mu\alpha.c_1 \parallel \tilde{\mu}x.c_2 \rangle$ where both rules $(\mu)$ and $(\tilde{\mu})$ can be applied at the same time, producing two different results. For example,

$$\langle \mu\alpha.\langle y \parallel \beta \rangle \parallel \tilde{\mu}x.\langle z \parallel \gamma \rangle \rangle \rightarrow_{\mu} \langle y \parallel \beta \rangle$$
$$\langle \mu\alpha.\langle y \parallel \beta \rangle \parallel \tilde{\mu}x.\langle z \parallel \gamma \rangle \rangle \rightarrow_{\tilde{\mu}} \langle z \parallel \gamma \rangle$$

Hence, the calculus is not confluent. But if the priority is given to one of the rules, we obtain two confluent subcalculi. In other words, there are two possible reduction strategies in the calculus that depend on the orientation of the critical pair. If the priority is given to $(\mu)$ redexes, call-by-value reduction is obtained, whereas giving the priority to $(\tilde{\mu})$ redexes simulates call-by-name reduction.

It is also possible to give two reduction rules for $\eta$ reduction:

$$\begin{array}{lrcll} (\eta_{\mu}) & \mu\alpha.\langle v \parallel \alpha \rangle & \rightarrow & v & \text{if } \alpha \notin fv(v) \\ (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x \parallel e \rangle & \rightarrow & e & \text{if } x \notin fv(e) \end{array}$$

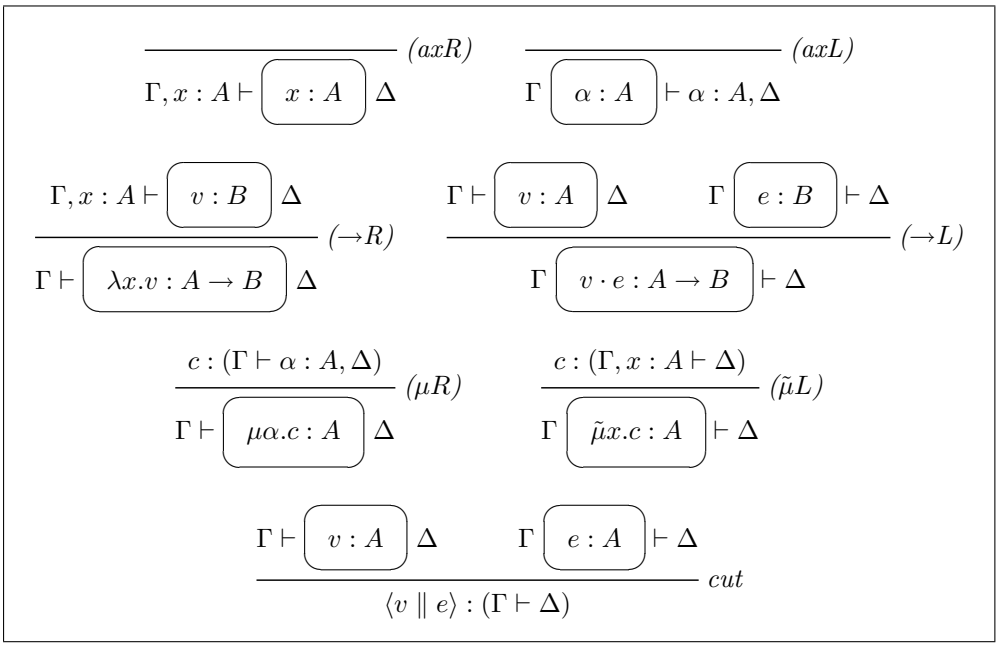



Figure 4.12: The type system for $\bar{\lambda}\mu\tilde{\mu}$-calculus

**The type system** $\bar{\lambda}\mu\tilde{\mu}$-terms correspond to proofs in a system denoted by $LK_{\mu\tilde{\mu}}$. This is a particular formulation of classical Gentzen's system LK, as one formula in a sequent can be distinguished. Therefore, besides regular sequents $\Gamma \vdash \Delta$, there are two other kinds of sequents, which have one distinguishable formula either in the premise, or in the conclusion: [3]

$$\Gamma\ \boxed{A} \vdash \Delta \text{ and } \Gamma \vdash \boxed{A}\ \Delta$$

This actually determines where the computation had happend and where it will continue.

*Simple types* of the form $A, B ::= p \mid A \to B$ are assigned to the terms of the calculus, where $p$ ranges over type variables. The type assignment system for the $\bar{\lambda}\mu\tilde{\mu}$-calculus is given in Figure 4.12. The relation between the $\bar{\lambda}\mu\tilde{\mu}$-calculus and ${}^*\mathcal{X}$ calculus is studied in Section 6.2 (page 89).

[3] Here a distinguished formula is denoted by putting an oval box around it.

# Chapter 5

# Erasure and duplication: the $^*\mathcal{X}$ calculus

In this chapter we give the syntax and reduction rules of the untyped $^*\mathcal{X}$ calculus, followed by the basic operational properties and the definition of typing rules. Although it is presented here as a counterpart of the implicative segment of the sequent calculus for classical logic, the system can naturally be extended to encompass other connectives as well. This extension is presented in Section 5.6.

## 5.1 The syntax

Terms are built from *names*. This concept differs essentially from the one applied in $\lambda$-calculus, where *variable* is the basic notion. The difference lies in the fact that a variable can be substituted by an arbitrary term, while a name can be only *renamed* (that is, substituted by another name). In our calculus the renaming is explicit, which means that it is expressed within the language itself and is not defined in the meta-theory. The reader will notice the presence of hats on some names. This notation has been borrowed from *Principia Mathematica* [WR25] and is used to denote the binding of a name.

**Definition 2 ($^*\mathcal{X}$-syntax)** *The syntax of $^*\mathcal{X}$-calculus is presented in Figure 5.1, where $x, y, z...$ range over an infinite set of innames and $\alpha, \beta, \gamma...$ range over an infinite set of outnames.*

| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $\langle x.\alpha \rangle$ | *capsule* |
| | $\mid$ | $\widehat{x}\, P\, \widehat{\beta} \cdot \alpha$ | *exporter* |
| | $\mid$ | $P\, \widehat{\alpha}\ [x]\ \widehat{y}\, Q$ | *importer* |
| | $\mid$ | $P \widehat{\alpha}\ \dagger\ \widehat{x} Q$ | *cut* |
| | $\mid$ | $x \odot P$ | *left-eraser* |
| | $\mid$ | $P \odot \alpha$ | *right-eraser* |
| | $\mid$ | $z{<}\frac{\widehat{x}}{\widehat{y}}\langle P]$ | *left-duplicator* |
| | $\mid$ | $[P\rangle \frac{\widehat{\alpha}}{\widehat{\beta}}{>}\gamma$ | *right-duplicator* |

Figure 5.1: The syntax of $^*\mathcal{X}$

**Elements of syntax**

There are eight constructors introduced in the syntax. To start with, we may distinguish the ground element (a *capsule*) from the higher order ones (that is, compound ones - the rest of the terms).

$\underline{\langle x.\alpha \rangle}$ : A *capsule* - is the only ground term, as it can not be decomposed into smaller elements. This is the starting component to build any term. It codes the axiom of the sequent calculus. It has one free inname $x$ and one free outname $\alpha$.[1]

$\underline{\widehat{y}\, P\, \widehat{\beta} \cdot \alpha}$ : An *exporter* - is a higher order term, obtained by binding an inname $y$ and an outname $\beta$ of the subterm $P$ and exporting this functionality on the new outname $\alpha$. An exporter corresponds to the right arrow introduction rule of the sequent calculus (see Figure 5.10 on page 61).

$\underline{P\, \widehat{\beta}\ [x]\ \widehat{y}\, Q}$ : An *importer* - is a term built from two subterms $P$ and $Q$ by binding an outname $\beta$ of the first and an inname $y$ of the second and making this functionality available on the free inname $z$. An importer corresponds to a left arrow introduction rule in the sequent calculus. The interaction between an exporter and an importer is specific and leads to one of the most important reduction rules in the calculus.

$\underline{P \widehat{\alpha}\ \dagger\ \widehat{x} Q}$ : A *cut* - is a term which originates from the cut rule in the sequent calculus. A cut is constructed by binding an outname $\alpha$ of a term $P$ and an inname $\beta$ of a term $Q$, as shown by the syntax. The process of computation will go towards eliminating cuts from terms.

$\underline{x \odot P}$ : *Left eraser* - is obtained by adding a fresh inname $x$ to an arbitrary term P. This term will be used to implement one of two (symmetric)

[1] See Definition 3 on page 39 for free innames and free outnames.

kinds of erasure in the calculus. Left-eraser codes left-weakening of the sequent calculus (see Figure 5.10 on page 61).

$\underline{P \odot \alpha}$ : $\underline{\textit{Right eraser}}$ - is constructed by adding a fresh outname $\alpha$ to an arbitrary term $P$. Its use is in implementing the other kind of erasure in the calculus. Right eraser originates from the right-weakening rule of the sequent calculus.

$\underline{z{<}^{\widehat{x}}_{\widehat{y}}\langle P]}$ : $\underline{\textit{Left duplicator}}$ - is used to implement duplication, i.e., one of the two symmetric kinds of duplication. It is obtained by binding the two innames $x, y$ of the term $P$ and introducing a single fresh free inname $z$. Left duplicator originates from the left-contraction rule (see Figure 5.10 on page 61).

$\underline{[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma}$ : $\underline{\textit{Right duplicator}}$ - is used to implement the second kind of duplication in the calculus. It is obtained by binding the two outnames $\alpha, \beta$ of the term $P$ and introducing a fresh free name $\gamma$. Right duplicator is inspired by the right-contraction rule.

### 5.1.1 Free and bound names

Names can be free or bound.

**Definition 3 (Free Names)** *The sets of* free innames *and* free outnames *are defined in Figure 5.2.*

| $S$ | $I(S)$ | $O(S)$ |
|---|---|---|
| $\langle x.\alpha\rangle$ | $x$ | $\alpha$ |
| $\widehat{x}\,P\,\widehat{\beta}\cdot\alpha$ | $I(P)\backslash\{x\}$ | $(O(P)\backslash\{\beta\})\cup\{\alpha\}$ |
| $P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q$ | $I(P)\cup(I(Q)\backslash\{y\})\cup\{x\}$ | $(O(P)\backslash\{\alpha\})\cup O(Q)$ |
| $P\widehat{\alpha}\;\dagger\;\widehat{x}Q$ | $I(P)\cup(I(Q)\backslash\{x\})$ | $(O(P)\backslash\{\alpha\})\cup O(Q)$ |
| $x\odot P$ | $I(P)\cup\{x\}$ | $O(P)$ |
| $P\odot\alpha$ | $I(P)$ | $O(P)\cup\{\alpha\}$ |
| $x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P]$ | $(I(P)\backslash\{x_1,\ x_2\})\cup\{x\}$ | $O(P)$ |
| $[P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha$ | $I(P)$ | $(O(P)\backslash\{\alpha_1,\ \alpha_2\})\cup\{\alpha\}$ |

Figure 5.2: Free names

*We write $N(P)$ for the* set *of free names of $P$, $I(P)$ the* sets *of free innames of $P$, and $O(P)$ the set of free outnames of $P$. Thus $N(P)$ =*

$I(P) \cup O(P)$. *A name which occurs in $P$ and which is not free is called a* bound name.

Notice that some term-constructors bind two names. It can be either two innames, two outnames or an inname and an outname. Moreover, these names sometimes belong to different subterms, as in the case of an exporter, or a cut. Independently from [UB01a], multiple binders were also proposed by [CH00] for dealing with the right implication rule.

### 5.1.2 Definitions and notations

The notation introduced in [KL07] to describe the λlxr reduction rules is rather complex (although reduction rules are rather simple), due to manipulation with lists of free variables when erasing and duplicating terms. This also incorporates the need for renaming operation. In the case of $^*\mathcal{X}$ calculus, it gets even more complex, due to the fact that we use names of two kinds, in contrast to the use of only variables in λlxr. Moreover, erasure and duplication come in two symmetric forms, which requires additional care.

**Sets and lists of free names** The notation $I(P)$, $O(P)$ and $N(P)$ is used to define sets of free innames, free outnames and free names of $P$, respectively. If we wish to see those sets as lists, we use the following notation: $\mathcal{I}^P$, $\mathcal{O}^P$ and $\mathcal{N}^P$, respectively. It is sometimes needed to use $\overline{I}(P)$ instead of $\mathcal{I}^P$, and similarly for outnames $\overline{O}(P)$, and names in general $\overline{N}(P)$. The bar is used to denote that we see the given set of names as a list, according to the total order which can be defined for the set of names. To exclude a name from a list, say outname $\alpha$, we write $\mathcal{O}^P \backslash \alpha$.

To denote a set of free innames in common for terms $P, Q$ we use $I(P \cap Q)$. Similarly for the free innames occurring in $P$ but not in $Q$ we use $I(P \backslash Q)$, the union of free innames is $I(P \cup Q)$. Similarly for names and outnames. In terms of lists, the notation used is the following: $\mathcal{I}^{P \cap Q}, \mathcal{I}^{P \cup Q}, \mathcal{I}^{P \backslash Q}$ for lists of free innames, and similarly for names and outnames.

The concatenation of lists $\mathcal{I}^P$ and $\mathcal{I}^Q$ is simply $\mathcal{I}^P, \mathcal{I}^Q$. To denote the binding of all names in one list, we use simply $\widehat{\mathcal{I}}$, $\widehat{\mathcal{O}}$.

**Indexing** Indexing is in fact a renaming operation, defined on free names, and used as side mechanism when implementing reduction rules.[2] We have

[2]Indexing is primarily used to implement duplication, as it will be presented in subsection 5.2.2.

chosen to use indexing, denoted *ind*, to make the syntax as simple as possible. We define indexing of free names for terms, lists and sets. For example $ind(P, x, 1)$ means that the free name $x$ in $P$ is renamed by $x_1$. This operation is primarily introduced to manipulate sets and lists of names, so $P_i = ind(P, N(P), i)$ means that $P_i$ is obtained by indexing free names in $P$ by index $i$, where $i \in N$. Simple notation $P_i$ for cases such as this one will be used when possible. We assume that indexing always creates fresh names. As we use it indexing preserves linearity.

*Lists* of free names can be indexed. We use $\mathcal{I}_i\,(\mathcal{O}_i, \mathcal{N}_i)$ to denote that all innames (outnames, names) in the list $\mathcal{I}\,(\mathcal{O}, \mathcal{N})$ are indexed by $i$. Accordingly, to denote an indexed list (with index $i$) of free innames (outnames, names) *of a term* $P$ we use: $\mathcal{I}_i^P\,(\mathcal{O}_i^P, \mathcal{N}_i^P)$. Similarly for *sets* of free names.

**Renaming** We define a renaming operation . It is a meta-operation which stands for the replacing of one free name in a term by another free name. Unlike in $\lambda$-calculus where a variable can be substituted by an arbitrary term, in $\mathcal{X}$-like calculi one can only replace a name by another name (Section 6.3.2 shows how explicit substitution of $\lambda$-calculus can be encoded in ${}^*\mathcal{X}$). Let $y$ be a free name in $P$, then $P\{x/y\}$ denotes the *renaming* in $P$ of $y$ by a fresh name $x$.

If the free innames of $P$ are given by $\mathcal{I}^P = (x, y, ...)$, and $\mathcal{I}_1^P = (x_1, y_1, ...)$ is an indexed list, then $P\{\mathcal{I}_1/\mathcal{I}\}$ denotes the term $P$ in which all free innames were simultaneously substituted by their indexed (by an index 1) variant. This notation allows us to build more subtle expressions. For example, given the terms $P, Q$ we may construct $P\{\mathcal{I}_1^{P\cap Q}/\mathcal{I}^{P\cap Q}\}$, to denote $P$ in which only the free innames occurring in both, $P$ and $Q$, are indexed by an index 1. Similar combinations are possible for outnames and names, with respect to also $P \cup Q$ and $P \backslash Q$.

**Convention on names** We adopt a convention on names, which is a Barendregt-style convention as defined in [Bar84]. We will consider that "a name is never both bound and free in the same term". We also assume that terms are defined up to $\alpha$-conversion, that is, that the renaming of bound innames or outnames does not change them.

**Modules** We define the notion of a *module* based on the definition of terms. A module is a part of a term (not a subterm) which can be permeated through the structure of that term (and its subterms) during the computation. This is defined by the so-called propagation rules. In some

sense it resembles the explicit substitution in the intuitionistic logic. There are two symmetric forms of modules, namely *left-module* and *right-module*. They are of the form:

$$\widehat{\alpha} \nearrow \widehat{x}Q \quad \text{and} \quad P\widehat{\beta} \nwarrow \widehat{y}$$

We say that $\alpha$ and $y$ are the *handles* of $\widehat{\alpha} \nearrow \widehat{x}Q$ and $P\widehat{\beta} \nwarrow \widehat{y}$, respectively. We say that two *modules are independent* if the handle of one module does not bind a free name inside the other module, and vice-versa, as follows:

| independent modules | condition |
|---|---|
| $\widehat{\alpha} \nearrow \widehat{x}Q,\ \widehat{\beta} \nearrow \widehat{y}R$ | $\alpha \notin N(R),\ \beta \notin N(Q)$ |
| $P\widehat{\alpha} \nwarrow \widehat{x},\ \ Q\widehat{\beta} \nwarrow \widehat{y}$ | $x \notin N(Q),\ y \notin N(P)$ |
| $P\widehat{\alpha} \nwarrow \widehat{x},\ \ \widehat{\beta} \nearrow \widehat{y}R$ | $x \notin N(R),\ \beta \notin N(P)$ |

We extend the definition of independent modules to a set of modules. We say that $M$ is a *set of independent modules* if every pair of modules in $M$ is independent.

### 5.1.3 Linearity

In the $^*\mathcal{X}$ calculus we consider only linear terms. We say that a term is *linear* if it satisfies the following:

- Every name has at most one free occurrence, and
- Every binder does bind an actual occurrence of a name (and therefore only one)

Linearity can be formally represented.

**Definition 4 (Linear terms)** *Formal definition of* linear terms *is given in Figure 5.3.*

Although the $^*\mathcal{X}$-syntax in general produces non-linear terms, every non-linear term can be translated into a linear one. Going from non-linear to linear proofs as we define them, using duplicators and erasers, is natural. This simply means that we analyze a proof, and if there are implicit contractions or weakenings, we make them explicit in a way that does not change the meaning of the original proof.

We present the transformation from non-linear to linear terms in Section 6.1. This is because the study of the relation between non-linear and

$$\frac{}{\langle x.\alpha\rangle \ \ linear}$$

$$\frac{P \ \ linear, \ \ x,\beta \in N(P), \ \ \alpha \notin N(P)}{\widehat{x}\, P\, \widehat{\beta}\cdot\alpha \ \ linear}$$

$$\frac{P,Q \ \ linear, \ \ \alpha \in N(P), \ \ x \in N(Q), \ \ y \notin N(P,Q), \ \ N(P)\cap N(Q)=\emptyset}{P\,\widehat{\alpha}\ [y]\ \widehat{x}\,Q \ \ linear}$$

$$\frac{P,Q \ \ linear, \ \ \alpha \in N(P), \ \ x \in N(Q), \ \ N(P)\cap N(Q)=\emptyset}{P\widehat{\alpha}\ \dagger\ \widehat{x}Q \ \ linear}$$

$$\frac{P \ \ linear, \ \ x \notin N(P)}{x \odot P \ \ linear} \qquad \frac{P \ \ linear, \ \ \alpha \notin N(P)}{P \odot \alpha \ \ linear}$$

$$\frac{P \ \ linear, \ \ x,y \in N(P), \ \ z \notin N(P)}{z{<}\frac{\widehat{x}}{\widehat{y}}\langle P] \ \ linear} \qquad \frac{P \ \ linear, \ \ \alpha,\beta \in N(P), \ \ \gamma \notin N(P)}{[P\rangle_{\widehat{\beta}}^{\widehat{\alpha}}{>}\gamma \ \ linear}$$

Figure 5.3: Linear terms

linear terms is in fact the study of the relation between the $\mathcal{X}$ calculus and the ${}^*\mathcal{X}$ calculus.

Here we give an illustration using the examples:

$$\langle x.\alpha\rangle \odot \alpha,$$

which is not linear since the outname $\alpha$ has two free occurrences. It can be represented in ${}^*\mathcal{X}$ by the term

$$[\langle x.\alpha_1\rangle \odot \alpha_2\rangle_{\widehat{\alpha_2}}^{\widehat{a_1}}{>}\alpha$$

(notice the role of a duplicator). The other typical case of non-linearity is when we have a binder which binds no free name, as for example, in the term

$$\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma$$

This is solved by using a linear term

$$\widehat{x}\,(\langle x.\alpha\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma$$

instead (notice the role of an eraser).

Thus, the linearity is achieved thanks to the presence of terms representing weakening and contraction. In some sense they are used to mimic non-linear terms; duplicator - when there are multiple occurrences of a name, and eraser - when there is a binder but the free name is missing, i.e., a binder does not bind anything.

A cut is an operation which binds two names, one inname and one outname. Names that are involved in a cut will be frequently mentioned and therefore the following definition.

**Definition 5 (Cut names)** *In a term $P\widehat{\alpha} \dagger \widehat{x}Q$ names $x$ and $\alpha$ are called* cut names*.*

**Definition 6 (Principal name)** *We define the* principal name *of a term $Q$ in the following way:*

| $Q$ | *principal names of $Q$* |
|---|---|
| $\langle x.\alpha \rangle$ | $x, \alpha$ |
| $\widehat{x}\, M\, \widehat{\beta} \cdot \alpha$ | $\alpha$ |
| $M\, \widehat{\alpha}\ [x]\ \widehat{y}\, N$ | $x$ |
| $M\widehat{\alpha} \dagger \widehat{x}N$ | *none* |
| $x \odot M$ | $x$ |
| $M \odot \alpha$ | $\alpha$ |
| $x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle M]$ | $x$ |
| $[M\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}{>}\alpha$ | $\alpha$ |

Notice that most of the terms have only one principal name, with exceptions of cut and capsule, which have zero and two, respectively.

**Remark** Sometimes it is convenient to say: a term *introduces*[3] the name, instead of: the name is principal for a term.

**Definition 7 (Logical and structural names)** *We say that a name is* logical *if it is introduced by either: a capsule, an importer or an exporter. We say that a name is* structural *if it is introduced by either an eraser or a duplicator.*[4]

[3] And therefore *freshly* introduces, due to linearity.

[4] Depending on whether a term codes logical or a structural inference rule.

Thus a principal name can be also logical or structural and therefore we have the following definition. This is important because it will be essential to distinguish those two kinds of principal names in the reduction procedure.

**Definition 8 (L-principal and S-principal names)** *We say that the name is L-principal (S-principal) for the term $P$ if it is a logical name (a structural name) and principal name for $P$.*

**Lemma 9** *Every term has at least a free logical outname.*

**Proof:** The proof goes by routine induction on the structure of terms. □.

**Remark** This would be no longer true if we were to extend the system with negation.

**Definition 10 (Contexts)** Contexts *are formally defined as follows:*

$$
\begin{array}{lllll}
C\{\ \} & ::= & \{\ \} & | & \widehat{x}\{\ \}\widehat{\beta}\cdot\alpha \\
 & | & \{\ \}\widehat{\alpha}\ [x]\ \widehat{y}Q & | & P\,\widehat{\alpha}\ [x]\ \widehat{y}\{\ \} \\
 & | & \{\ \}\widehat{\alpha}\ \dagger\ \widehat{x}Q & | & P\widehat{\alpha}\ \dagger\ \widehat{x}\{\ \} \\
 & | & x\odot\{\ \} & | & \{\ \}\odot\alpha \\
 & | & z{<}\frac{\widehat{x}}{\widehat{y}}\langle\{\ \}] & | & [\{\ \}\rangle_{\widehat{\beta}}^{\widehat{\alpha}}{>}\gamma \\
 & | & C\{C\{\ \}\} & &
\end{array}
$$

**Remark** A context is a term with a hole in which another term can be placed. Therefore $C\{P\}$ denotes placing the term $P$ in the context $C\{\ \}$.

**Remark** We use $P = Q$ to denote that the terms $P$ and $Q$ are syntactically equal.

**Definition 11 (Subterm relation $\preccurlyeq$)** *A term $Q$ is a* subterm *of a term $P$, denoted as $Q \preccurlyeq P$ if there is a context $C\{\ \}$ such that $P = C\{Q\}$.*

**Lemma 12** *The* subterm *relation is reflexive, antisymmetric and transitive (i.e., is an order):*

1. Reflexivity $P \preccurlyeq P$
2. Antisymmetry *If* $P \preccurlyeq Q$ *and* $Q \preccurlyeq P$ *then* $P = Q$
3. Transitivity*: If* $P \preccurlyeq Q$ *and* $Q \preccurlyeq R$ *then* $P \preccurlyeq R$

**Remark** The symbol = stands for syntactic equality.

**Proof:**

1. The first point is straightforward. If $P \preccurlyeq P$, then by the subterm definition we have $\exists\ C\{\ \}$ such that $P = C\{P\}$. This stands if we choose $C\{\ \}$ to be $\{\ \}$.

2. Let $P \preccurlyeq Q$ and $Q \preccurlyeq R$. By definition $\exists\ C'\{\ \}, C''\{\ \}$ such that $C'\{P\} = Q$ and $C''\{Q\} = P$. From $C'\{C''\{Q\}\} = Q$ we derive $C'\{\ \} = C''\{\ \} = \{\ \}$. Finally we can conclude $P = Q$.

3. On the one hand, from $P \preccurlyeq Q$ by definition we have: $\exists\ C'\{\ \}$ such that $C'\{P\} = Q$. On the other hand, from $Q \preccurlyeq R$ by definition we have: $\exists\ C''\{\ \}$ such that $C''\{Q\} = R$. Thus, $C''\{C'\{P\}\} = R$ and therefore by definition we have $P \preccurlyeq R$. □

The following definition introduces the notion of a simple context, i.e., a context which is not composed of other contexts. Notice that it is similar to the definition of context, with the exception of the cases $\{\ \}$ and $C\{C\{\ \}\}$ which are omitted.

**Definition 13 (Simple context)** *A context $C\{\ \}$ is called a* simple *context if $C\{\ \}$ is one of the following:*

| | | | | |
|---|---|---|---|---|
| $C\{\ \}$ | := | $\widehat{x}\{\ \}\widehat{\beta}\cdot\alpha$ | \| | |
| | \| | $\{\ \}\widehat{\alpha}\ [x]\ \widehat{y}\,Q$ | \| | $P\,\widehat{\alpha}\ [x]\ \widehat{y}\{\ \}$ |
| | \| | $\{\ \}\widehat{\alpha}\ \dagger\ \widehat{x}Q$ | \| | $P\widehat{\alpha}\ \dagger\ \widehat{x}\{\ \}$ |
| | \| | $x \odot \{\ \}$ | \| | $\{\ \} \odot \alpha$ |
| | \| | $z{<}\frac{\widehat{x}}{\widehat{y}}\langle\{\ \}]$ | \| | $[\{\ \}\rangle_{\widehat{\beta}}^{\widehat{\alpha}}{>}\gamma$ |

Using the definition of a simple context we will formulate the notion of immediate subterm as follows.

**Definition 14 (Immediate subterm)** *A term* Q *is an* immediate subterm *of* P *if $P = C\{Q\}$ and $C\{\ \}$ is a simple context.*

**Example 15** *A term can have either one, two, or zero immediate subterms. For example, $Q\,\widehat{\alpha}\ [x]\ \widehat{y}\,R$ has two immediate subterms (these are $P$ and $Q$), $\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha$ has one (a term $P$), whilst $\langle x.\alpha\rangle$ has zero immediate subterms.*

Using the definition of a simple context we will formulate the notion of immediate subterm as follows.

**Definition 16 (Immediate subterm)** *A term* Q *is an* immediate subterm *of* P *if* $P = C\{Q\}$ *and* $C\{\ \}$ *is a simple context.*

**Definition 17 (Context with two holes)** *Using the definition of a context* $C\{\ \}$*, we specify the notion of a* context with two holes *in the following way:*

$$\begin{array}{rcl} C\{\ ,\ \} & ::= & \{\ \}\,\widehat{\alpha}\,[x]\,\widehat{y}\{\ \} \quad | \quad \{\ \}\widehat{\alpha}\,\dagger\,\widehat{x}\{\ \} \\ & | & C\{C\{\ \}, C\{\ \}\} \quad | \quad C\{C\{\ ,\ \}\} \end{array}$$

Like for contexts (with one hole) we define the notion of a *simple context with two holes.*

**Definition 18 (Simple context with two holes)**

$$C\{\ ,\ \} \quad ::= \quad \{\ \}\,\widehat{\alpha}\,[x]\,\widehat{y}\{\ \} \quad | \quad \{\ \}\widehat{\alpha}\,\dagger\,\widehat{x}\{\ \}$$

Using this definition, the notion of *immediate subterm* can be naturally extended as to encompass the cases when we speak about *two immediate subterms.*

**Lemma 19** *The following holds:*

1. *If* $\alpha \in N(P)$ *then there exists a unique term* $Q \preccurlyeq P$ *such that* $\alpha$ *is a principal name for* $Q$*.*

2. *If* $x \in N(P)$ *then there exists a unique term* $R \preccurlyeq P$ *such that* $x$ *is a principal name for* $R$*.*

**Remark** We will use the notation $Q^{\alpha}$ to specify that $Q$ has $\alpha$ as a principal name. Similarly, we use $R^{x}$ to emphasize that $R$ has $x$ as a principal name.

**Proof:** We prove the first point. The proof goes by induction on the structure of a term $P$ and case analysis.

Let $\alpha \in P$.

- Case: $\alpha$ is a principal name for $P$. Then $Q = P$.

- Case: $\alpha$ is not a principal name for $P$. Then, either $P = C\{R\}$ or $P = C\{R_1, R_2\}$, where $R, R_1$ and $R_2$ denote immediate subterms of $P$.

- $P = C\{R\}$. By induction hypothesis, and since by linearity $\alpha \in R$, we have: $\exists Q \preccurlyeq R$ such that $\alpha$ is a principal name for $Q$. By using transitivity (lemma 12), from $Q \preccurlyeq R$ and $R \preccurlyeq P$ we infer $Q \preccurlyeq P$.
- $P = C\{R_1, R_2\}$. By the linearity condition we know that $\alpha$ belongs to either $N(R_1)$ or $N(R_2)$ (not to both). Thus we have two subcases, which correspond to the previous case. In the first case $C\{R_1, R_2\}$ is seen as $C'\{R_1\}$, where $C'\{\ \} = C\{\{\ \}, R_2\}$, and in the second case as $C''\{R_2\}$, where $C''\{\ \} = C\{R_1, \{\ \}\}$. Recall that $R_1, R_2$ are immediate subterms of $P$ by definition.

The second point of the lemma refers to innames instead of outnames, and the proof goes similarly. □

## 5.2 Reduction rules

In this section we define the reduction relation, $\rightarrow$. Like classical cut-elimination, this system has many rules. For convenience we split them into groups, which are themselves split into symmetrical "left" and "right" subgroup, with the exception of the logical group.

Reduction rules are grouped into

- Activation rules (left and right)
- Structural actions (left and right)
- Deactivation rules (left and right)
- Logical actions
- Propagation rules (left and right)

**Some abbreviations** We introduce some abbreviations in order to represent the reduction rules in a convenient form.

| instead of | we write |
|---|---|
| $x_1 \odot (\ldots (x_n \odot P)\ldots)$ | $x_1 \odot \ \ldots\ x_n \odot P$ |
| $(\ldots (P \odot \alpha_1)\ \ldots) \odot \alpha_n$ | $P \odot \alpha_1\ \ldots\ \odot \alpha_n$ |
| $x_1 < \frac{\widehat{y_1}}{\widehat{z_1}} \langle \ldots x_n < \frac{\widehat{y_n}}{\widehat{z_n}} \langle P \rangle \ldots \rangle$ | $(x_1, \ldots,\ x_n) < \frac{(\widehat{y_1}, \ldots, \widehat{y_n})}{(\widehat{z_1}, \ldots, \widehat{z_n})} \langle P \rangle$ |
| $[\ldots [P \rangle \frac{\widehat{\beta_1}}{\widehat{\gamma_1}} > \alpha_1 \ldots \rangle \frac{\widehat{\beta_n}}{\widehat{\gamma_n}} > \alpha_n$ | $[P \rangle \frac{(\widehat{\beta_1}, \ldots, \widehat{\beta_n})}{(\widehat{\gamma_1}, \ldots, \widehat{\gamma_n})} > (\alpha_1, \ldots, \alpha_n)$ |

**Congruence rules** At this point, we also assume some simple congruence rules on terms. We do not devote special attention to them here, since these rules represent a small fragment of what is studied in Part III. They originate from the sequent calculus where independent weakening and contraction rules can be permuted. Also we have the commutativity of bound names of a contraction, and an associativity of names when two contractions are combined in a specific way.

- Commuting the names inside of a duplicator:

$$\boxed{\begin{array}{lcl} x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle P] & \equiv & x<\frac{\widehat{x_2}}{\widehat{x_1}}\langle P] \\ [P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\alpha & \equiv & [P\rangle\frac{\widehat{\alpha_2}}{\widehat{\alpha_1}}>\alpha \end{array}}$$

- Permuting independent duplicators:

$$\boxed{\begin{array}{lcl} x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle y<\frac{\widehat{y_1}}{\widehat{y_2}}\langle P]] & \equiv & y<\frac{\widehat{y_1}}{\widehat{y_2}}\langle x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle P]] \\ [[P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\alpha\rangle\frac{\widehat{\beta_1}}{\widehat{\beta_2}}>\beta & \equiv & [[P\rangle\frac{\widehat{\beta_1}}{\widehat{\beta_2}}>\beta\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\alpha \\ [x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle P]\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\alpha & \equiv & x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle [P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\alpha] \end{array}}$$

  the conditions in the first rule are $y \notin \{x_1, x_2\}$ and $x \notin \{y_1, y_2\}$ and in the second $\beta \notin \{\alpha_1, \alpha_2\}$ and $\alpha \notin \{\beta_1, \beta_2\}$. These congruence rules say that, provided they are not related, duplicators can be freely commuted. The third rule allows us drop parenthesis and use a simplified notation

$$x <\frac{\widehat{x_1}}{\widehat{x_2}}\langle P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}} > \alpha$$

  and more generally

$$\mathcal{I} <\frac{\widehat{\mathcal{I}_1}}{\widehat{\mathcal{I}_2}}\langle P\rangle\frac{\widehat{\mathcal{O}_1}}{\widehat{\mathcal{O}_2}} > \mathcal{O}$$

  where $\mathcal{I}$ and $\mathcal{O}$ are lists of names. In the simple case when $\mathcal{I} = ()$, we write simply $[P\rangle\frac{\widehat{\mathcal{O}_1}}{\widehat{\mathcal{O}_2}}>\mathcal{O}$. The case when $\mathcal{O} = ()$ is not possible as stated by Lemma 9.

- When the names are triplicated, one can do it in any order:

$$\boxed{\begin{array}{lcl} z<\frac{\widehat{y}}{\widehat{x_3}}\langle y<\frac{\widehat{x_1}}{\widehat{x_2}}\langle P]] & \equiv & z<\frac{\widehat{x_1}}{\widehat{y}}\langle y<\frac{\widehat{x_2}}{\widehat{x_3}}\langle P]] \\ [[P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}>\beta\rangle\frac{\widehat{\beta}}{\widehat{\alpha_3}}>\gamma & \equiv & [[P\rangle\frac{\widehat{\alpha_2}}{\widehat{\alpha_3}}>\beta\rangle\frac{\widehat{\alpha_1}}{\widehat{\beta}}>\gamma \end{array}}$$

This can be seen as an associativity of names bound by a ternary duplicator.

- Permuting the erasers:

$$\boxed{\begin{array}{ccc} y \odot x \odot P & \equiv & x \odot y \odot P \\ P \odot \alpha \odot \beta & \equiv & P \odot \beta \odot \alpha \\ (x \odot P) \odot \alpha & \equiv & x \odot (P \odot \alpha) \end{array}}$$

The third rule suggests that we may drop parenthesis and write:

$$x \odot P \odot \alpha,$$

and more generally we may write:

$$\mathcal{I} \odot P \odot \mathcal{O}$$

We now present the reduction rules of $^*\mathcal{X}$ calculus.

### 5.2.1 Activation rules

We now present activation rules which describe the non-deterministic choice of classical cut-elimination. More precisely, during the process of cut-elimination sometimes we have to choose the left or the right subtree to push the cut further. This choice is captured by the activation rules. But before defining the reduction rules, we have to introduce new symbols for active cuts into the syntax.

**Definition 20 (Active Cuts)** *The syntax is extended with two* active cuts*:*

$$P, Q ::= \ldots \mid P\widehat{\alpha} \nearrow \widehat{x}Q \mid P\widehat{\alpha} \nwarrow \widehat{x}Q$$

A cut can be activated only towards an eraser or a duplicator. This means it can be left-activated only if one has either an eraser or a duplicator on the left, and similarly, it can be right-activated only if one has an eraser or a duplicator on the right. See Figure 5.4. Eraser and duplicator are terms which originate from structural inference rules of weakening and contraction. It is not possible to activate a cut towards the terms corresponding to logical inference rules, but in all other cases the activation is possible. Understanding of this group is easier if it is considered together with a dual group called deactivation rules. See Section 5.2.3 in this chapter.

Activation rules are a potential source of non-confluence, wich is illustrated by Example 21.

<u>*Left*</u> :

$(act\text{-}L)\ :\ P\widehat{\alpha} \dagger \widehat{x}Q \ \rightarrow\ P\widehat{\alpha} \nearrow \widehat{x}Q$, if $\alpha$ not L-principal for $P$

<u>*Right*</u> :

$(act\text{-}R)\ :\ P\widehat{\alpha} \dagger \widehat{x}Q \ \rightarrow\ P\widehat{\alpha} \nwarrow \widehat{x}Q$, if $x$ not L-principal for $Q$

Figure 5.4: Activation rules

**Example 21** *Terms $P\widehat{\alpha} \nearrow \widehat{x}Q$ and $P\widehat{\alpha} \nwarrow \widehat{x}Q$ are essentially different. This becomes obvious if we take the example when both $\alpha$ and $x$ are introduced by erasers. Take*

$$P = M \odot \alpha \qquad \textit{and} \qquad Q = x \odot N,$$

*where $M$ and $N$ are arbitrary terms. Then we have:*

$$\begin{array}{l} (M \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}(x \odot N) \ \rightarrow\ \mathcal{I}^{N\setminus x} \odot M \odot \mathcal{O}^{N} \\ (M \odot \alpha)\widehat{\alpha} \nwarrow \widehat{x}(x \odot N) \ \rightarrow\ \mathcal{I}^{M} \odot N \odot \mathcal{O}^{M\setminus \alpha} \end{array}$$

This simple example is equivalent to the one pointed out by Lafont [GLT89]. It will be revisited in a diagrammatic framework (in Section 8.2.1 on page 108).

**Remark** By constantly giving priority to either left or right activation, we may remove the non-confluence from the calculus and thus obtain two confluent subcalculi. In the $\bar{\lambda}\mu\tilde{\mu}$-calculus, if one gives priority to one of two sides, then one obtains a call-by-name or a call-by-value calculus. We suspect in accordance to was noted for $\mathcal{X}$ (seen as Urban's calculus) in [Urb01], that this does not extend to ${}^{*}\mathcal{X}$.

### 5.2.2 Structural actions

Structural actions consist of four reduction rules, specifying *erasure* and *duplication* by referring to the situation when an active cut faces an eraser or a duplicator. Structural actions are given in Figure 5.5. These computational features were studied more in the framework of intuitionistic logic [DG01],[KL07].

### 5.2.3 Deactivation rules

Active cuts are introduced into the system by activation rules. There are other groups of rules dealing with active cuts; they can be propagated, dupli-

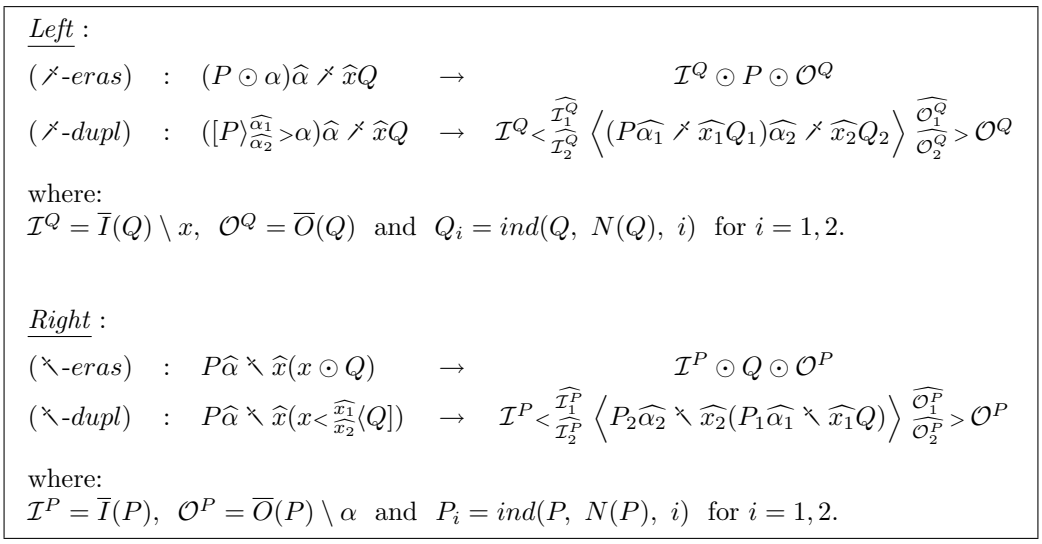



Figure 5.5: Structural actions

cated, erased or *deactivated*. In this subsection we speak about deactivation of previously active cuts. Deactivation rules are given in Figure 5.6.

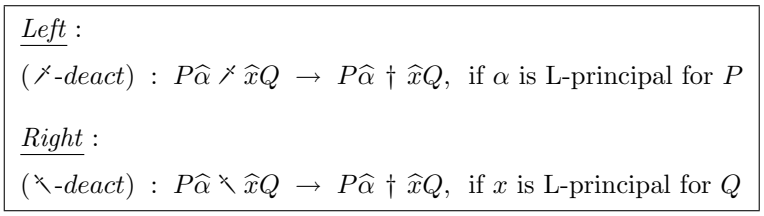



Figure 5.6: Deactivation rules

The deactivation is defined for terms originating from logical inference rules of the sequent calculus. That is, in case of a capsule or an exporter for left-deactivation and, in the case of capsule and importer for right-deactivation. There is no deactivation for an eraser and a duplicator.

There exists a *duality* between activation and deactivation rules. One can be obtained from the other by reversing the arrow, and negating the side conditions. The design of activation and deactivation rules is such that it does not enable looping, that is, the side conditions do not allow activation followed by deactivation of a cut, or vice versa.

It is possible to define deactivation more explicitly, removing the side

conditions:

$$
\begin{array}{lll}
\underline{\textit{Left}}: & & \\
(\nearrow\textit{-deact}_1) & : \langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R & \rightarrow \langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R \\
(\nearrow\textit{-deact}_2) & : (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R \\
 & & \\
\underline{\textit{Right}}: & & \\
(\nwarrow\textit{-deact}_1) & : P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle & \rightarrow P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \\
(\nwarrow\textit{-deact}_2) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R) & \rightarrow P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R)
\end{array}
$$

### 5.2.4 Logical actions

The purpose of logical actions is to define reduction when L-principal names are involved in a cut. See Figure 5.7.

$$
\begin{array}{lll}
(\textit{ren-L}) & : \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}Q & \rightarrow Q\{y/x\} \\
(\textit{ren-R}) & : P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle & \rightarrow P\{\beta/\alpha\} \\
(\textit{ei-insert}) & : (\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R) & \rightarrow \textit{either} \left\{ \begin{array}{l} (Q\widehat{\gamma} \dagger \widehat{y}P)\widehat{\beta} \dagger \widehat{z}R \\ Q\widehat{\gamma} \dagger \widehat{y}(P\widehat{\beta} \dagger \widehat{z}R) \end{array} \right.
\end{array}
$$

Figure 5.7: Logical actions

First two logical rules define *merging* of a capsule with another term by using the renaming operation. The operation $\{y/x\}$ is a meta operation, which resembles the $\lambda$-calculus meta-substitution. But here it simply denotes rewriting an occurrence of a free name (unique by linearity) by another name, which means that it does not essentially change the term.

The merging of a capsule with another element is defined following Urban's local cut-elimination procedure (page 50, [Urb00]), whereas a different approach was taken in [vBLL05] (see logical rules in Section 2) and in [Len03] (see logical rules in Section 4). It may seem that merging of a capsule with another term can be simulated by rules which do not involve meta operation [Len03, vBLL05], but in general this is not true since propagation over active cuts is not enabled (see the next section for propagation rules).

The third logical action describes the direct interaction between an exporter and an importer, which results in *inserting* the (immediate) subterm of an exporter between the two (immediate) subterms of an importer.

### 5.2.5 Propagation rules

Propagation rules describe the propagation of a cut through the structure of terms. This is a step-by-step propagation (the reduction rules "describe" propagation). It is important to note that propagation of a cut over another inactive cut is possible, which allows an elegant representation of $\beta$-reduction. The rules are divided into "left" and "right" symmetric groups, see Figures 5.8 and 5.9.

$$
\begin{array}{lcll}
(exp\nearrow\text{-}prop) & : & (\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ \widehat{x}\,(P\widehat{\beta}\nearrow\widehat{y}R)\,\widehat{\gamma}\cdot\alpha, \quad \alpha\neq\beta \\
\\
(imp\nearrow\text{-}prop_1) & : & (P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q, \quad \beta\in O(P) \\
(imp\nearrow\text{-}prop_2) & : & (P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\,\widehat{\alpha}\,[x]\,\widehat{z}\,(Q\widehat{\beta}\nearrow\widehat{y}R), \quad \beta\in O(Q) \\
\\
(cut(c)\nearrow\text{-}prop) & : & (P\widehat{\alpha}\,\dagger\,\widehat{x}\langle x.\beta\rangle)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\widehat{\alpha}\,\dagger\,\widehat{y}R \\
(cut\nearrow\text{-}prop_1) & : & (P\widehat{\alpha}\,\dagger\,\widehat{x}Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\widehat{\alpha}\,\dagger\,\widehat{x}Q, \quad \beta\in O(P),\ Q\neq\langle x.\beta\rangle \\
(cut\nearrow\text{-}prop_2) & : & (P\widehat{\alpha}\,\dagger\,\widehat{x}Q)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ P\widehat{\alpha}\,\dagger\,\widehat{x}(Q\widehat{\beta}\nearrow\widehat{y}R), \quad \beta\in O(Q),\ Q\neq\langle x.\beta\rangle \\
\\
(L\text{-}eras\nearrow\text{-}prop) & : & (x\odot P)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ x\odot(P\widehat{\beta}\nearrow\widehat{y}R) \\
(R\text{-}eras\nearrow\text{-}prop) & : & (P\odot\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ (P\widehat{\beta}\nearrow\widehat{y}R)\odot\alpha, \quad \alpha\neq\beta \\
\\
(L\text{-}dupl\nearrow\text{-}prop) & : & (x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\rangle)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\widehat{\beta}\nearrow\widehat{y}R] \\
(R\text{-}dupl\nearrow\text{-}prop) & : & ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\beta}\nearrow\widehat{y}R & \rightarrow\ [P\widehat{\beta}\nearrow\widehat{y}R\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha, \quad \alpha\neq\beta
\end{array}
$$

Figure 5.8: Left propagation

Observe for example the first rule in the left group. The rule is denoted as $(exp\nearrow -prop)$ and it shows how an active cut (in fact, a module $\widehat{\beta}\nearrow\widehat{y}R$) enters from the right-hand side through an exporter, up to its immediate subterm. The rules which define propagation over an exporter or a cut require some side conditions to decide to which of the two immediate subterms the module will go.

The rules which require additional explanations are $(cut(c)\nearrow\text{-}prop)$ and $(\nwarrow cut(c)\text{-}prop)$. These are the rules which define an exception when performing propagation rules. They handle the case of propagation over a cut with a capsule whose both names are cut-names. If we exclude these rules from the system, we could construct an infinite reduction sequence.

$$
\begin{array}{lll}
(\nwarrow exp\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(\widehat{y}\,Q\,\widehat{\beta}\cdot\gamma) & \rightarrow\ \widehat{y}\,(P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\cdot\gamma \\
 & & \\
(\nwarrow imp\text{-}prop_1) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\ [y]\ \widehat{z}\,R) & \rightarrow\ (P\widehat{\alpha} \nwarrow \widehat{x}Q)\,\widehat{\beta}\ [y]\ \widehat{z}\,R, \quad x \in I(Q) \\
(\nwarrow imp\text{-}prop_2) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\ [y]\ \widehat{z}\,R) & \rightarrow\ Q\,\widehat{\beta}\ [y]\ \widehat{z}\,(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad x \in I(R) \\
 & & \\
(\nwarrow cut(c)\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R) & \rightarrow\ P\widehat{\alpha} \dagger \widehat{y}R \\
(\nwarrow cut\text{-}prop_1) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \rightarrow\ (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R, \quad x \in I(Q),\ Q \neq \langle x.\beta\rangle \\
(\nwarrow cut\text{-}prop_2) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) & \rightarrow\ Q\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}R), \quad x \in I(R),\ Q \neq \langle x.\beta\rangle \\
 & & \\
(\nwarrow L\text{-}eras\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(y \odot Q) & \rightarrow\ y \odot (P\widehat{\alpha} \nwarrow \widehat{x}Q), \quad x \neq y \\
(\nwarrow R\text{-}eras\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q \odot \beta) & \rightarrow\ (P\widehat{\alpha} \nwarrow \widehat{x}Q) \odot \beta \\
 & & \\
(\nwarrow L\text{-}dupl\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(y_{<\frac{\widehat{y_1}}{\widehat{y_2}}}\langle Q]) & \rightarrow\ y_{<\frac{\widehat{y_1}}{\widehat{y_2}}}\langle P\widehat{\alpha} \nwarrow \widehat{x}Q], \quad x \neq y \\
(\nwarrow R\text{-}dupl\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}([Q\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}>\beta}) & \rightarrow\ [P\widehat{\alpha} \nwarrow \widehat{x}Q\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}>\beta}
\end{array}
$$

Figure 5.9: Right propagation

**Example 22** *An example of an infinite reduction sequence in absence of* $(cut(c)\nearrow\text{-}prop)$ *and* $(\nwarrow cut(c)\text{-}prop)$ *rules:*

$$
\begin{array}{ll}
 & (P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \dagger \widehat{y}R \\
\rightarrow & (P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R \\
\rightarrow & P\widehat{\alpha} \dagger \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R) \\
\rightarrow & P\widehat{\alpha} \dagger \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R) \\
\rightarrow & P\widehat{\alpha} \nwarrow \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R) \\
\rightarrow & (P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \dagger \widehat{y}R \\
\rightarrow & (P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \dagger \widehat{y}R
\end{array}
$$

Besides that, the solution offered is intuitive as we would expect the terms

$$(P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R \ \text{ and } \ P\widehat{\alpha} \nwarrow \widehat{x}(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R)$$

to reduce to the same term (which is in this case $P\widehat{\alpha} \dagger \widehat{y}R$).

### Structural rules and convergence

Let us have a close look at the structural rules of duplication. For example the rule ($\nearrow$-*dupl*) :

$$([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}<\alpha)\widehat{\alpha} \nearrow \widehat{x}Q \quad \rightarrow \quad \mathcal{I}^Q{<}^{\widehat{\mathcal{I}_1^Q}}_{\widehat{\mathcal{I}_2^Q}} \left\langle (P\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2 \right\rangle {}^{\widehat{\mathcal{O}_1^Q}}_{\widehat{\mathcal{O}_2^Q}}{>}\mathcal{O}^Q$$

yields the term $(P\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2$ in the context of a certain number of left and right contractions. Notice that this term contains two left modules, namely

$$\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1 \quad \text{and} \quad \widehat{\alpha_2} \nearrow \widehat{x_2}Q_2,$$

which are independent since $\alpha_1, \alpha_2 \in P$ (and therefore $\alpha_2 \notin Q_1,\ \alpha_1 \notin Q_2$). One may wonder whether the order of modules is relevant, that is, whether the two terms

$$S_1 = (P\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2 \quad \text{and} \quad S_2 = (P\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2)\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1$$

should be considered the same. Intuitively, in a given situation, there is no reason to prefer using one term over another.

**Definition 23 (Convergent reductions)** *We say that the two terms $P$ and $Q$ have* convergent reductions *if they share the set of normal forms, denoted*

$$P =_{NF} Q$$

In the following theorem we prove that $S_1$ and $S_2$ (see above) can be consided to be equivalent, in a sense that they share the same set of normal forms, i.e., they have *convergent* reductions, that is:

$$S_1 =_{NF} S_2$$

**Theorem 24 (Convergence)** *Let $P,\ Q,\ P_1, \dots P_n,\ Q_1, \dots Q_n$ be arbitrary terms and $\phi$ an arbitrary permutation of $(1, ...n)$, then*

$(a)$ $\quad (\dots(P\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\dots)\widehat{\alpha_n} \nearrow \widehat{x_n}Q_n \ =_{NF}$
$(\dots(P\widehat{\alpha_{\phi(1)}} \nearrow \widehat{x_{\phi(1)}}Q_{\phi(1)})\dots)\widehat{\alpha_{\phi(n)}} \nearrow \widehat{x_{\phi(n)}}Q_{\phi(n)}$

*where $\{\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1, \dots \widehat{\alpha_n} \nearrow \widehat{x_n}Q_n\}$ is a set of independent modules.*

$(b)$ $\quad P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}(\dots(P_n\widehat{\alpha_n} \nwarrow \widehat{x_n}Q)\dots) \ =_{NF}$
$P_{\phi(1)}\widehat{\alpha_{\phi(1)}} \nwarrow \widehat{x_{\phi(1)}}(\dots(P_{\phi(n)}\widehat{\alpha_{\phi(n)}} \nwarrow \widehat{x_{\phi(n)}}Q)\dots)$

*where $\{P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}, \dots P_n\widehat{\alpha_n} \nwarrow \widehat{x_n}\}$ is a set of independent modules.*

**Proof:** The parts $(a)$ and $(b)$ of the property can be shown by case analysis and induction on the structure of terms $P$ and $Q$, respectively. The technique is similar and therefore it suffices to prove one of them; in this case we prove $(a)$. Thus the proof goes by case analysis and induction on the structure of $P$. We assume that the property holds for any subterm $P' \preccurlyeq P$. For simplicity reasons we analyze the variant with two modules instead of looking at the case for $n$ in general.

$$(P\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R \quad =_{NF} \quad (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \nearrow \widehat{x}Q$$

Remark that the side condition is $\alpha, \beta \in N(P)$. Due to linearity it follows that $\alpha \notin N(R)$ and $\beta \notin N(Q)$. Moreover it follows that $P$ is not a capsule and $Q \neq \langle x.\beta \rangle$, which are the conditions relevant for the proof.

(1) Case: neither $\alpha$ nor $\beta$ are principal for $P$.
There are two cases to consider here, depending whether $P$ has one immediate subterm, $P'$, or it has two, $P'$ and $P''$.

- $P = C[P']$ : this case obviously follows by induction.
- $P = C[P', P'']$ : if both names, $\alpha, \beta$, belong to the same subterm, this case is equivalent to the previous. Otherwise, if they belong to different subterms, for example $\alpha \in P', \beta \in P''$ then, on the one hand we have
  $((C\{P', P''\})\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R$
  $\rightarrow (C\{P'\widehat{\alpha} \nearrow \widehat{x}Q, P''\})\widehat{\beta} \nearrow \widehat{y}R$
  $\rightarrow C\{P'\widehat{\alpha} \nearrow \widehat{x}Q, P''\widehat{\beta} \nearrow \widehat{y}R\}$ and on the other hand
  $((C\{P', P''\})\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \nearrow \widehat{x}Q$
  $\rightarrow (C\{P', P''\widehat{\beta} \nearrow \widehat{y}R\})\widehat{\alpha} \nearrow \widehat{x}Q$
  $\rightarrow C\{P'\widehat{\alpha} \nearrow \widehat{x}Q, P''\widehat{\beta} \nearrow \widehat{y}R\}$.
  We may notice that the resulting terms are syntactically equal.

(2) Case: $\alpha$ is L-principal for $P$ (yields $\beta$ is not principal for $P$)
On one hand we have
$(P\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}Z \rightarrow (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R,\ Q \neq \langle x.\beta \rangle$
$\rightarrow (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q \rightarrow (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \nearrow \widehat{x}Q$.
This is exactly the term on the right-hand side of the $=_{NF}$ sign.

(3) Case: $\beta$ is L-principal for $P$ (yields $\alpha$ is not principal for $P$)
The proof goes similarly as for the previous case, only here the term on the right-hand side of the $=_{NF}$ reduces to the term on the left-hand side.

(4) Case: $\alpha$ is S-principal for $P$ (yields $\beta$ is not principal for $P$)
There are two possibilities: either $P$ is an eraser, or $P$ is a duplicator.

- $P$ is an eraser $P = P' \odot \alpha$
The reductions are the following. On the one hand we have
$((P' \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R$
$\rightarrow (\mathcal{I}^{Q\backslash x} \odot P' \odot \mathcal{O}^Q)\widehat{\beta} \nearrow \widehat{y}R, \ \ \beta \in N(P')$
$\rightarrow^+ \mathcal{I}^{Q\backslash x} \odot (P'\widehat{\beta} \nearrow \widehat{y}R) \odot \mathcal{O}^Q.$
On the other hand:
$((P' \odot \alpha)\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \nearrow \widehat{x}Q \rightarrow^+ ((P'\widehat{\beta} \nearrow \widehat{y}R) \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}Q$
$\rightarrow \mathcal{I}^{Q\backslash x} \odot (P'\widehat{\beta} \nearrow \widehat{y}R) \odot \mathcal{O}^Q$
- $P$ is a duplicator $P = [P'\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha$
On the one hand we have
$(([P'\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\alpha} \nearrow \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R$
$\rightarrow (\mathcal{I}^{Q\backslash x} {<}^{\widehat{\mathcal{I}_1}^{Q\backslash x}}_{\widehat{\mathcal{I}_2}^{Q\backslash x}}\langle(P'\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2\rangle^{\widehat{\mathcal{O}_1}^Q}_{\widehat{\mathcal{O}_2}^Q}{>}\mathcal{O}^Q)\widehat{\beta} \nearrow \widehat{y}R,$
$\beta \in N(P')$
$\rightarrow^+ \mathcal{I}^{Q\backslash x} {<}^{\widehat{\mathcal{I}_1}^{Q\backslash x}}_{\widehat{\mathcal{I}_2}^{Q\backslash x}}\langle((P'\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2)\widehat{\beta} \nearrow \widehat{y}R\rangle^{\widehat{\mathcal{O}_1}^Q}_{\widehat{\mathcal{O}_2}^Q}{>}\mathcal{O}^Q$
On the other hand:
$(([P'\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \nearrow \widehat{x}Q$
$\rightarrow ([P'\widehat{\beta} \nearrow \widehat{y}R\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\alpha} \nearrow \widehat{x}Q$
$\rightarrow \mathcal{I}^{Q\backslash x} {<}^{\widehat{\mathcal{I}_1}^{Q\backslash x}}_{\widehat{\mathcal{I}_2}^{Q\backslash x}}\langle((P'\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2\rangle^{\widehat{\mathcal{O}_1}^Q}_{\widehat{\mathcal{O}_2}^Q}{>}\mathcal{O}^Q.$
By induction the two are convergent.

(5) Case: $\beta$ is S-principal for $P$ (yields $\alpha$ is not principal for $P$)
The proof for this case goes similarly to the previous one.

Thus we are done with the proof. □

## 5.3 Operational properties

The reduction system enjoys some desirable properties as expressed by the following lemma.

**Theorem 25 (Basic properties of $\rightarrow$)**

1. *Preservation of* free names *(interface preservation):*
$$\textit{If } P \rightarrow Q \textit{ then } N(P) = N(Q)$$

2. *Preservation of* linearity*:*
$$\textit{If } P \textit{ is linear and } P \rightarrow Q \textit{ then } Q \textit{ is linear}$$

**Proof:** These properties can be confirmed by checking carefully each rule. □

Free names preservation holds in ${}^*\mathcal{X}$ due to the use of erasers and duplicators in rewrite rules (there is a similar situation in $\lambda$lxr [KL07]). The situation is different in the $\lambda$-calculus and in the $\lambda$lxr-calculus where free variables may be lost during computation. This property is sometimes referred to as "interface preservation" as in the framework of *interaction nets* [Laf95b]. The preservation of linearity is a minimal requirement, since ${}^*\mathcal{X}$ has been designed as a linear model of computation.

**Simplification rules** We define the *simplification rules*, denoted $\dashrightarrow$, which can be seen as an efficient way to simplify terms. They are not reduction rules as they do not involve cuts. The point is that applying a duplicator to an eraser is of no interest and can be avoided by using simplification rules, as defined by:

$$\boxed{\begin{array}{lcll} (s_L) & : & x{<}\frac{\widehat{y}}{\widehat{z}}\langle z \odot P] & \dashrightarrow \; P\{x/y\} \\ (s_R) & : & [P \odot \gamma\rangle\frac{\widehat{\beta}}{\widehat{\gamma}}{>}\alpha & \dashrightarrow \; P\{\alpha/\beta\} \end{array}}$$

They are applied before the reduction rules, that is, we give them high priority during computation. One can see them as a kind of garbage collection, as they simplify computation by preventing the situation when we duplicate a term to erase one or both copies in the next step.

It is easy to see that the simplification rules preserve the set of free names, linearity and types. The rules can be given in a more general way:

$$\boxed{\begin{array}{lcll} (s_L^g) & : & \mathcal{I}{<}\frac{\widehat{\mathcal{I}_1}}{\widehat{\mathcal{I}_2}}\langle \mathcal{I}_2 \odot P] & \dashrightarrow \; P\{\mathcal{I}/\mathcal{I}_1\} \\ (s_R^g) & : & [P \odot \mathcal{O}_2\rangle\frac{\widehat{\mathcal{O}_1}}{\widehat{\mathcal{O}_2}}{>}\mathcal{O} & \dashrightarrow \; P\{\mathcal{O}/\mathcal{O}_1\} \end{array}}$$

## 5.4 The type assignment system

We only consider here linear terms to which we will add type information. Given a set $T$ of basic types, a type is given by

$$A, B ::= T \mid A \to B.$$

In order to assign terms to the sequent proofs of the system given in Figure 3.3 (page 20), we will introduce expressions

$$P \vdots\ \Gamma \vdash \Delta$$

which represent the *type assignment*[5], where $P$ is an $^*\mathcal{X}$-term, $\Gamma$ is a context (antecedent) whose domain consists of free innames of $P$ and $\Delta$ is a context (succedent) whose domain consists of free outnames of $P$. Comma in the expression $\Gamma, \Delta$ stands for the set union. For example $\Gamma$ as a set of type declarations for innames could be $x : A,\ y : B$, while $\Delta$ as a set of declarations for outnames could be $\alpha : A$, $\beta : A \to B$, $\gamma : C$.

We will say that a term $P$ is *typable* if there exist contexts $\Gamma$ and $\Delta$ such that $P \vdots\ \Gamma \vdash \Delta$ holds in the system of inference rules given by Figure 5.10.

If we remove the term decoration in the type system of Figure 5.10, we get the classical sequent calculus as presented in Figure 3.1 (page 18).

**Example 26** *We show here how the types are assigned to a term which codes the Peirce's law.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x.\alpha_1\rangle \vdots\ x : A \vdash \alpha_1 : A}\,(\text{ax})}{\langle x.\alpha_1\rangle \odot \beta \vdots\ x : A \vdash \alpha_1 : A, \beta : B}\,(\text{weak-R})}{\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma \vdots\ \vdash \alpha_1 : A, \gamma : A \to B}\,(\to\text{R}) \quad \dfrac{}{\langle y.\alpha_2\rangle \vdots\ y : A \vdash \alpha_2 : A}\,(\text{ax})}{(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle \vdots\ z : (A \to B) \to A \vdash \alpha_1 : A, \alpha_2 : A}\,(\to\text{L})}{[(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha \vdots\ z : (A \to B) \to A \vdash \alpha : A}\,(\text{cont-R})}{\widehat{z}\,([(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\,\widehat{\alpha}\cdot\delta \vdots\ \vdash \delta : ((A \to B) \to A) \to A}\,(\to\text{R})$$

This example will be revisited in the framework of diagrams (page 117).

[5]Technically we assign contexts, which are sets of pairs (name, formula), to terms. If we forget about labels and consider only types, we are going back to the system $G1$ where contexts are multisets of formulas.

$$\frac{}{\langle x.\alpha\rangle :\cdot \;\; x :A \vdash \alpha :A}\ (ax)$$

$$\frac{P :\cdot \;\; \Gamma \vdash \alpha :A, \Delta \qquad Q :\cdot \;\; \Gamma', y :B \vdash \Delta'}{P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q :\cdot \;\; \Gamma, \Gamma', x :A \to B \vdash \Delta, \Delta'}\ (L\to) \qquad \frac{P :\cdot \;\; \Gamma, x :A \vdash \alpha :B, \Delta}{\widehat{x}\,P\,\widehat{\alpha}\cdot\beta :\cdot \;\; \Gamma \vdash \beta :A \to B, \Delta}\ (R\to)$$

$$\frac{P :\cdot \;\; \Gamma \vdash \alpha :A, \Delta \qquad Q :\cdot \;\; \Gamma', x :A \vdash \Delta'}{P\widehat{\alpha} \dagger \widehat{x}Q :\cdot \;\; \Gamma, \Gamma' \vdash \Delta, \Delta'}\ (cut)$$

$$\frac{P :\cdot \;\; \Gamma \vdash \Delta}{x \odot P :\cdot \;\; \Gamma, x :A \vdash \Delta}\ (weak\text{-}L) \qquad \frac{P :\cdot \;\; \Gamma \vdash \Delta}{P \odot \alpha :\cdot \;\; \Gamma \vdash \alpha :A, \Delta}\ (weak\text{-}R)$$

$$\frac{P :\cdot \;\; \Gamma, x :A, y :A \vdash \Delta}{z{<}\frac{\widehat{x}}{\widehat{y}}\langle P] :\cdot \;\; \Gamma, z :A \vdash \Delta}\ (cont\text{-}L) \qquad \frac{P :\cdot \;\; \Gamma \vdash \alpha :A, \beta : A, \Delta}{[P\rangle_{\widehat{\beta}}^{\widehat{\alpha}}{>}\gamma :\cdot \;\; \Gamma \vdash \gamma :A, \Delta}\ (cont\text{-}R)$$

Figure 5.10: The type system for ${}^*\mathcal{X}$

**Example 27** *We show here how to assign contexts to the term which corresponds to $\lambda xyz.xz(yz)$ known as the S-combinator of $\lambda$-calculus. The proof-tree for the S-combinator is given at the end of this chapter, on page 77. The corresponding ${}^*\mathcal{X}$-term is:*[6]

$$\widehat{\omega}\,(\widehat{u}\,(\widehat{x}\,(x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle\langle x_2.\epsilon\rangle\,\widehat{\epsilon}\;[w]\;\widehat{v}\,\underline{(\langle\langle x_1.\delta\rangle\,\widehat{\delta}\;[u]\;\widehat{y}\,\langle y.\beta\rangle\rangle)\,\widehat{\beta}\;[v]\;\widehat{z}\,\langle z.\gamma\rangle\rangle])\,\widehat{\gamma}\cdot\eta)\,\widehat{\eta}\cdot\theta)\,\widehat{\theta}\cdot\alpha$$

In addition this example will be revisited in a diagrammatic framework (page 118).

### The witness reduction property

Since the beginning, the work on ${}^*\mathcal{X}$ calculus is strongly related to the sequent calculus. In that sense, if we want to see computation as proof-transformation, the property of witness reduction is essential.

We first formulate the lemma which proves the admissibility of renaming.

**Lemma 28** *The following rules are admissible in the typing system of the ${}^*\mathcal{X}$ calculus:*

[6] Some parts of terms are underlined to make the reading easier.

$$\frac{P \vdots\ \Gamma, x : A \vdash \Delta}{P\{y/x\} \vdots\ \Gamma, y : A \vdash \Delta}\,(R\text{-}l) \qquad \frac{P \vdots\ \Gamma \vdash \alpha : A, \Delta}{P\{\beta/\alpha\} \vdots\ \Gamma \vdash \beta : A, \Delta}\,(R\text{-}r)$$

**Proof:** The admissibility of rules is proved by induction on the structure of terms. We give the proof for $R$-$r$. In the simple case we have

$$\frac{\langle x.\alpha\rangle \vdots\ x : A \vdash \alpha : A}{\langle x.\alpha\rangle\{\beta/\alpha\} \vdots\ x : A \vdash \beta : A}\,(R\text{-}r)$$

since by definition of renaming $\langle x.\alpha\rangle\{\beta/\alpha\} = \langle x.\beta\rangle$. It is easy to check that this also stands for other terms who have $\alpha$ as a principal name, namely for $P = \widehat{x}\, Q\, \widehat{\gamma} \cdot \alpha$, $P = [Q)^{\widehat{\alpha_1}}_{\widehat{\alpha_1}} > \alpha$ and $P = Q \odot \alpha$, using the condition of linearity which guarantees that $\alpha \notin N(Q)$.

On the other hand, if $\alpha$ is not a principal name for $P$, then by Lemma 19, there exists a context $C\{\ \}$ such that $P = C\{Q\}$ where $\alpha$ is principal for $Q$. By linearity condition we have $C\{Q\}\{\beta/\alpha\} = C\{Q\{\beta/\alpha\}\}$. Thus, by induction, the rule

$$\frac{P \vdots\ \Gamma \vdash \alpha : A, \Delta}{P\{\beta/\alpha\} \vdots\ \Gamma \vdash \beta : A, \Delta}\,(R\text{-}r)$$

is admissible. □

This can be generalized to accommodate renaming of multiple names.

**Lemma 29** *The following rule is admissible:*

$$\frac{P \vdots\ \Gamma, \Gamma' \vdash \Delta, \Delta'}{P\{dom(\Gamma'_1)/d(\Gamma')\}\{dom(\Delta'_1)/dom(\Delta')\} \vdots\ \Gamma, \Gamma'_1 \vdash \Delta, \Delta'_1}\,(Rs)$$

*where* $\Gamma'_1 = ind(\Gamma', 1)$ *and* $\Delta'_1 = ind(\Delta', 1)$.

**Proof:** Follows directly from the previous lemma.

The following lemma shows the derivability of rules which are used in the proof of witness reduction theorem.

**Lemma 30** *The following rules are derivable in the typing system of the* $^*\mathcal{X}$ *calculus:*

$$\frac{P \vdots\ \Gamma \vdash \Delta}{dom(\Gamma') \odot P \odot dom(\Delta') \vdots\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(Ws)$$

$$\frac{P \vdots\ \Gamma, \Gamma'_1, \Gamma'_2 \vdash \Delta, \Delta'_1, \Delta'_2}{dom(\Gamma') < \widehat{dom(\Gamma'_1)}\ \widehat{dom(\Gamma'_2)} \langle P \rangle^{\widehat{dom(\Delta'_1)}}_{\widehat{dom(\Delta'_2)}} > dom(\Delta') \vdots\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(Cs)$$

**Proof:** By induction on the cardinals of $dom(\Gamma')$ and $dom(\Delta')$. □

**Theorem 31 (Witness reduction)** *Let $S$ be an ${}^*\mathcal{X}$-term and $\Gamma, \Delta$ contexts. Then the following holds:*

$$\textit{If } S :\cdot\ \Gamma \vdash \Delta \textit{ and } S \to S', \textit{ then } S' :\cdot\ \Gamma \vdash \Delta$$

**Remark** The linearity and free names of $S$ are preserved (Theorem 25).

**Proof:** The proof is straightforward and goes by inspecting the reduction rules, and by induction on the structure of terms. We first write the typing derivation for the term on the left-hand side of the reduction rule, and then for the term on the right-hand side.

**Logical actions:**

• Take the $(ren\text{-}L)$ rule: $\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}Q \ \to\ Q\{y/x\}$.
On the one hand we have:

$$\dfrac{\dfrac{}{\boxed{\langle y.\alpha\rangle} :\cdot\ y : A \vdash \alpha : A}\,(ax) \qquad \boxed{Q} :\cdot\ \Gamma, x : A \vdash \Delta}{\boxed{\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}Q} :\cdot\ \Gamma, y : A \vdash \Delta}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{Q} :\cdot\ \Gamma, x : A \vdash \Delta}{\boxed{Q\{y/x\}} :\cdot\ \Gamma, y : A \vdash \Delta}\,(R\text{-}l)$$

• Take the $(ren\text{-}R)$ rule: $P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \ \to\ P\{\beta/\alpha\}$.
On the one hand we have:

$$\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{}{\boxed{\langle x.\beta\rangle} :\cdot\ x : A \vdash \beta : A}\,(ax)}{\boxed{P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle} :\cdot\ \Gamma \vdash \beta : A, \Delta}\,(cut)$$

On the other hand,

$$\frac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta}{\boxed{P\{\beta/\alpha\}} :\cdot\ \Gamma \vdash \beta : A, \Delta}\ (R\text{-}r)$$

• Take the *(ei-insert)* rule: $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R) \ \rightarrow\ \text{either}\begin{cases}(Q\widehat{\gamma}\dagger\widehat{y}P)\widehat{\beta}\dagger\widehat{z}R\\ Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)\end{cases}$,
with the condition $y, \beta \in N(P)$.
On the one hand, for $S$ we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma, y : A \vdash \beta : B, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\alpha} :\cdot\ \Gamma \vdash \alpha : A \to B, \Delta}\ (\to R) \quad \dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma' \vdash \gamma : A, \Delta' \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', z : B \vdash \Delta''}{\boxed{Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R} :\cdot\ \Gamma', \Gamma'', x : A \to B \vdash \Delta', \Delta''}\ (\to L)}{\boxed{(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R)} :\cdot\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\ (cut)$$

On the other hand, for $S'$, this time we have to consider two cases because the rule has two possible results. The first case is:

$$\dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma' \vdash \gamma : A, \Delta' \qquad \dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma, y : A \vdash \beta : B, \Delta \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', z : B \vdash \Delta''}{\boxed{P\widehat{\beta}\dagger\widehat{z}R} :\cdot\ \Gamma, \Gamma'', y : A \vdash \Delta, \Delta''}\ (cut)}{\boxed{Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)} :\cdot\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\ (cut)$$

The second case is:

$$\dfrac{\dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma' \vdash \gamma : A, \Delta' \qquad \boxed{\mathrm{P}} :\cdot\ \Gamma, y : A \vdash \beta : B, \Delta}{\boxed{Q\widehat{\gamma}\dagger\widehat{y}P} :\cdot\ \Gamma', \Gamma \vdash \beta : B, \Delta', \Delta}\ (cut) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', z : B \vdash \Delta''}{\boxed{(Q\widehat{\gamma}\dagger\widehat{y}P)\widehat{\beta}\dagger\widehat{z}R} :\cdot\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\ (cut)$$

**Structural actions - left subgroup:**

• Take the rule *(⊀-eras)*: $(P \odot \beta)\widehat{\beta} \nprec \widehat{y}R \ \rightarrow\ \mathcal{I}^R \odot P \odot \mathcal{O}^R$.
On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \Delta}{\boxed{P \odot \beta} :\cdot\ \Gamma \vdash \beta : B, \Delta}\,(\textit{weak-R}) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma', y : B \vdash \Delta'}{\boxed{(P \odot \beta)\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\textit{cut})$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \Delta}{\boxed{\mathcal{I}^R \odot P \odot \mathcal{O}^R} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(Ws)$$

- Take the *($\nearrow$-dupl)* rule:

$$([P\rangle\widehat{\beta_2}>\beta)\widehat{\beta} \nearrow \widehat{y}Q \quad \rightarrow \quad \mathcal{I}^Q < \widehat{\mathcal{I}_2^Q} \left\langle (P\widehat{\beta_1} \nearrow \widehat{y_1}Q_1)\widehat{\beta_2} \nearrow \widehat{y_2}Q_2 \right\rangle \widehat{\mathcal{O}_2^Q} > \mathcal{O}^Q,$$

with $\mathcal{I}^Q = \overline{I}(Q) \setminus y$ and $\mathcal{O}^Q = \overline{O}(Q)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \beta_1 : B, \beta_2 : B, \Delta}{\boxed{[P\rangle\widehat{\beta_2}>b} :\cdot\ \Gamma' \vdash \beta : B, \Delta'}\,(\textit{cont-R}) \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma, y : B \vdash \Delta'}{\boxed{([P\rangle\widehat{\beta_2}>\beta)\widehat{\beta} \nearrow \widehat{y}Q} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\textit{cut})$$

On the other hand,

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \beta_1 : B, \beta_2 : B, \Delta \qquad \dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma', y : B \vdash \Delta'}{\boxed{Q_1} :\cdot\ \Gamma_1', y_1 : B \vdash \Delta_1'}\,(Rs)}{\boxed{P\widehat{\beta_1} \nearrow \widehat{y_1}Q_1} :\cdot\ \Gamma, \Gamma_1' \vdash \beta_2 : B, \Delta, \Delta_1'}\,(\textit{cut}) \qquad \dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma', y : B \vdash \Delta'}{\boxed{Q_2} :\cdot\ \Gamma_2', y_2 : B \vdash \Delta_2'}\,(Rs)}{\dfrac{\boxed{(P\widehat{\beta_1} \nearrow \widehat{y_1}Q_1)\widehat{\beta_2} \nearrow \widehat{y_2}Q_2} :\cdot\ \Gamma, \Gamma_1', \Gamma_2' \vdash \Delta, \Delta_1', \Delta_2'}{\boxed{\mathcal{I}^Q < \widehat{\mathcal{I}_2^Q} \left\langle (P\widehat{\beta_1} \nearrow \widehat{y_1}Q_1)\widehat{\beta_2} \nearrow \widehat{y_2}Q_2 \right\rangle \widehat{\mathcal{O}_2^Q} > \mathcal{O}^Q} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(Cs)}\,(\textit{cut})$$

where $dom(\Gamma) = \mathcal{I}(P)$ and $dom(\Delta) = \mathcal{O}(P) \setminus \alpha$.

**Structural actions -right subgroup:**

• Take the rule *($\nwarrow$-eras)*: $P\widehat{\alpha} \nwarrow \widehat{x}(Q \odot x) \;\; \rightarrow \;\; \mathcal{I}^P \odot Q \odot \mathcal{O}^P$.
On the one hand we have:

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma' \vdash \Delta'}{\boxed{Q \odot x} :\cdot\ \Gamma', x : A \vdash \Delta'}\,(L\text{-}weak)}{\boxed{P\widehat{\alpha} \nwarrow \widehat{x}(Q \odot x)} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma' \vdash \Delta'}{\boxed{\mathcal{I}^P \odot Q \odot \mathcal{O}^P} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(weak)$$

• Take the right-duplication rule *($\nwarrow$-dupl)*:
$P\widehat{\alpha} \nwarrow \widehat{x}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle Q]) \;\; \rightarrow \;\; \mathcal{I}^P {<}^{\widehat{\mathcal{I}_1^P}}_{\widehat{\mathcal{I}_2^P}} \left\langle P_2\widehat{\alpha_2} \nwarrow \widehat{x_2}(P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}Q) \right\rangle {}^{\widehat{\mathcal{O}_1^P}}_{\widehat{\mathcal{O}_2^P}}{>} \mathcal{O}^P,$
with $\mathcal{I}^P = \overline{I}(P)$ and $\mathcal{O}^P = \overline{O}(P) \setminus \alpha$. On the one hand we have:

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{\boxed{\mathrm{Q}} :\cdot\ \Gamma', x_1 : A, x_2 : A \vdash \Delta'}{\boxed{x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle Q]} :\cdot\ \Gamma', x : A \vdash \Delta'}\,(cont\text{-}L)}{\boxed{P\widehat{\alpha} \nwarrow \widehat{x}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle Q])} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta}{\boxed{P_2} :\cdot\ \Gamma_2 \vdash \alpha_2 : A, \Delta_2}\,(Rs) \qquad \dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta}{\boxed{P_1} :\cdot\ \Gamma_1 \vdash \alpha_1 : A, \Delta_1}\,(Rs) \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma', x_1 : A, x_2 : A \vdash \Delta'}{\boxed{P_1\widehat{\alpha_1} \nearrow \widehat{x_1}Q} :\cdot\ \Gamma_1, \Gamma' \vdash \Delta_1, \Delta'}\,(cut)}{\dfrac{\boxed{P_2\widehat{\alpha_2} \nwarrow \widehat{x_2}(P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}Q)} :\cdot\ \Gamma_2, \Gamma_1, \Gamma' \vdash \Delta_2, \Delta_1, \Delta'}{\boxed{\mathcal{I}^P {<}^{\widehat{\mathcal{I}_1^P}}_{\widehat{\mathcal{I}_2^P}} \left\langle P_2\widehat{\alpha_2} \nwarrow \widehat{x_2}(P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}Q) \right\rangle {}^{\widehat{\mathcal{O}_1^P}}_{\widehat{\mathcal{O}_2^P}}{>} \mathcal{O}^P} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(Cs)}\,(cut$$

where $dom(\Gamma) = \mathcal{I}(P)$ and $dom(\Delta) = \mathcal{O}(P) \setminus \alpha$.

**Propagation rules - left subgroup:**

• Take the *(exp↗-prop)* rule: $(\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R \;\;\rightarrow\;\; \widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha$, where $\alpha \neq \beta$, and by linearity $\beta \in N(P)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma, x:A \vdash \gamma : B, \beta : C, \Delta}{\boxed{\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha} :\cdot\ \Gamma \vdash \alpha : A \to B, \beta : C, \Delta}\,(\to R) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma', y : C \vdash \Delta'}{\boxed{(\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma' \vdash \alpha : A \to B, \Delta, \Delta'}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma, x:A \vdash \gamma : B, \beta : C, \Delta \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma', y : C \vdash \Delta'}{\dfrac{\boxed{P\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma', x:A \vdash \gamma : B, \Delta, \Delta'}{\boxed{\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha} :\cdot\ \Gamma, \Gamma' \vdash \alpha : A \to B, \Delta, \Delta'}\,(\to R)}\,(cut)$$

• Take the *(imp↗-prop$_1$)* rule: $(P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta} \nearrow \widehat{y}R \;\;\rightarrow\;\; (P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q$, when $\beta \in P$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \beta : C, \Delta \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma', z : B \vdash \Delta'}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q} :\cdot\ \Gamma, \Gamma', x : A \to B \vdash \beta : C, \Delta, \Delta'}\,(\to L) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', y : C \vdash \Delta''}{\boxed{(P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B \vdash \Delta, \Delta', \Delta''}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \beta : C, \Delta \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', y : C \vdash \Delta''}{\dfrac{\boxed{P\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma', x : A \to B \vdash \beta : C, \Delta, \Delta' \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma', z : B \vdash \Delta'}{\boxed{(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B \vdash \Delta, \Delta', \Delta''}\,(\to L)}\,(cut)$$

• For the other rule *(imp↗-prop$_2$)*: $(P\,\widehat{\alpha}\,[x]\,\widehat{z}\,Q)\widehat{\beta} \nearrow \widehat{y}R \;\;\rightarrow\;\; P\,\widehat{\alpha}\,[x]\,\widehat{z}\,(Q\widehat{\beta} \nearrow \widehat{y}R)$, when $\beta \in Q$, the proof proceeds likewise .

• Take the *(cut(c)↗-prop)* rule: $(P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R \;\;\rightarrow\;\; P\widehat{\alpha} \dagger \widehat{y}R$.
On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{}{\boxed{\langle x.\beta\rangle} :\cdot\ x : A \vdash \beta : A}\,(ax)}{\boxed{P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle} :\cdot\ \Gamma \vdash \beta : A, \Delta}\,(cut) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma', y : A \vdash \Delta'}{\boxed{(P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma', y : A \vdash \Delta'}{\boxed{P\widehat{\alpha} \dagger \widehat{y}R} :\cdot\ \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(cut)$$

• Take the *(cut↗-prop$_1$)* rule: $(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R \ \rightarrow\ (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q$, with $\beta \in N(P)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \beta : B, \Delta \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma', x : A \vdash \Delta'}{\boxed{P\widehat{\alpha} \dagger \widehat{x}Q} :\cdot\ \Gamma, \Gamma' \vdash \beta : B, \Delta, \Delta'}\,(cut) \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', y : B \vdash \Delta''}{\boxed{(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\,(cut)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{\mathrm{P}} :\cdot\ \Gamma \vdash \alpha : A, \beta : B, \Delta \qquad \boxed{\mathrm{R}} :\cdot\ \Gamma'', y : B \vdash \Delta''}{\boxed{P\widehat{\beta} \nearrow \widehat{y}R} :\cdot\ \Gamma, \Gamma' \vdash \beta : B, \Delta, \Delta'}\,(cut) \qquad \boxed{\mathrm{Q}} :\cdot\ \Gamma', x : A \vdash \Delta'}{\boxed{(P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q} :\cdot\ \Gamma, \Gamma', \Gamma'' \vdash \Delta, \Delta', \Delta''}\,(cut)$$

• For *(cut↗-prop$_2$)* rule: $(P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R \ \rightarrow\ P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \nearrow \widehat{y}R)$, when $\beta \in Q$, the proof is similar.

• Take the *(L-eras↗-prop)* rule: $(x \odot P)\widehat{\beta} \nearrow \widehat{y}R \ \rightarrow\ x \odot (P\widehat{\beta} \nearrow \widehat{y}R)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \;\vdots\; \Gamma, \beta : B \vdash \Delta}{\boxed{x \odot P} \;\vdots\; \Gamma, x : A \vdash \beta : B, \Delta}\,(\textit{weak-L}) \qquad \boxed{\mathrm{R}} \;\vdots\; \Gamma', y : B \vdash \Delta'}{\boxed{(x \odot P)\widehat{\beta} \nearrow \widehat{y}R} \;\vdots\; \Gamma, \Gamma', x : A \vdash \Delta, \Delta'}\,(\textit{cut})$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} \;\vdots\; \Gamma, \beta : B \vdash \Delta \qquad \boxed{\mathrm{R}} \;\vdots\; \Gamma', y : B \vdash \Delta'}{\dfrac{\boxed{P\widehat{\beta} \nearrow \widehat{y}R} \;\vdots\; \Gamma, \Gamma' \vdash \Delta, \Delta'}{\boxed{x \odot (P\widehat{\beta} \nearrow \widehat{y}R)} \;\vdots\; \Gamma, \Gamma', x : A \vdash \Delta, \Delta'}\,(\textit{weak-L})}\,(\textit{cut})$$

• For *(R-eras↗-prop)* rule: $(P \odot \alpha)\widehat{\beta} \nearrow \widehat{y}R \;\; \rightarrow \;\; (P\widehat{\beta} \nearrow \widehat{y}R) \odot \alpha$, with $\alpha \neq \beta$, the proof is similar.

• Take *(L-dupl↗-prop)* rule: $(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P])\widehat{\beta} \nearrow \widehat{y}R \;\; \rightarrow \;\; x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\widehat{\beta} \nearrow \widehat{y}R]$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{\mathrm{P}} \;\vdots\; \Gamma, x_1 : A, x_2 : A \vdash \beta : B, \Delta}{\boxed{x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P]} \;\vdots\; \Gamma, x : A \vdash \beta : B, \Delta}\,(\textit{contr-L}) \qquad \boxed{\mathrm{R}} \;\vdots\; \Gamma', y : B \vdash \Delta'}{\boxed{(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P])\widehat{\beta} \nearrow \widehat{y}R} \;\vdots\; \Gamma, \Gamma', x : A \vdash \Delta, \Delta'}\,(\textit{cut})$$

On the other hand,

$$\dfrac{\boxed{\mathrm{P}} \;\vdots\; \Gamma, x_1 : A, x_2 : A \vdash \beta : B, \Delta \qquad \boxed{\mathrm{R}} \;\vdots\; \Gamma', y : B \vdash \Delta'}{\dfrac{\boxed{P\widehat{\beta} \nearrow \widehat{y}R} \;\vdots\; \Gamma, \Gamma', x_1 : A, x_2 : A \vdash \Delta, \Delta'}{\boxed{x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\widehat{\beta} \nearrow \widehat{y}R]} \;\vdots\; \Gamma, \Gamma', x : A \vdash \Delta, \Delta'}\,(\textit{contr-L})}\,(\textit{cut})$$

• For *(R-dupl↗-prop)* rule: $([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\beta} \nearrow \widehat{y}R \;\; \rightarrow \;\; [P\widehat{\beta} \nearrow \widehat{y}R\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha$, with $\alpha \neq \beta$, the proof is similar to the previous case.

**Propagation rules - right subgroup:**
The proof goes similarly as for the left subgroup of propagation rules. □

**Theorem 32 (⇢ preserves types)** *Simplification rules preserve types.*

$$\textit{If } S :\cdot\ \Gamma \vdash \Delta \textit{ and } S \dashrightarrow S', \textit{ then } S' :\cdot\ \Gamma \vdash \Delta$$

**Proof:** By analyzing the proof trees corresponding to $S$ and $S'$, for both simplification rules.

• Observe the $(s_L)$ rule: $x{<}\overset{\widehat{y}}{\widehat{z}}\langle z \odot P] \dashrightarrow P\{x/y\}$. On the one hand, for $S$ we have:

$$\dfrac{\dfrac{\boxed{P} :\cdot\ \Gamma, y : A \vdash \Delta}{\boxed{z \odot P} :\cdot\ \Gamma, y : A, z : A \vdash \Delta}\,(L\text{-}weak)}{\boxed{x{<}\overset{\widehat{y}}{\widehat{z}}\langle z \odot P]} :\cdot\ \Gamma, x : A \vdash \Delta}\,(L\text{-}cont)$$

On the other hand, for $S'$ we have:

$$\dfrac{\boxed{P} :\cdot\ \Gamma, y : A \vdash \Delta}{\boxed{P\{x/y\}} :\cdot\ \Gamma, x : A \vdash \Delta}\,(R\text{-}l)$$

• Observe the $(s_R)$ rule: $[P \odot \gamma\rangle\overset{\widehat{\beta}}{\widehat{\gamma}}{>}\alpha \dashrightarrow P\{\alpha/\beta\}$. On the one hand, for $S$ we have:

$$\dfrac{\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \beta : A, \Delta}{\boxed{P \odot \gamma} :\cdot\ \Gamma \vdash \beta : A, \gamma : A, \Delta}\,(R\text{-}weak)}{\boxed{[P \odot \gamma\rangle\overset{\widehat{\beta}}{\widehat{\gamma}}{>}\alpha} :\cdot\ \Gamma \vdash \alpha : A, \Delta}\,(R\text{-}cont)$$

On the other hand, for $S'$ we have:

$$\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \beta : A, \Delta}{\boxed{P\{\alpha/\beta\}} :\cdot\ \Gamma \vdash \alpha : A, \Delta}\,(R\text{-}r)$$

□

## 5.5 Examples of implementations

In this section we show how some of the data types can be implemented in this calculus, which is needed if one wants to see ${}^*\mathcal{X}$ calculus as a programming language.

### 5.5.1 Booleans

We choose the terms for booleans *true* and *false* to be the following:

$$\begin{array}{rcl} true & \triangleq & \widehat{y}\,(\widehat{x}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta)\,\widehat{\beta}\cdot\gamma \\ false & \triangleq & \widehat{x}\,(\widehat{y}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta)\,\widehat{\beta}\cdot\gamma \end{array}$$

Then the term representing the conditional is a defined in a rather simple way

$$Cond\{\mathbf{B},P,Q\} \triangleq \boxed{\mathbf{B}}\widehat{\gamma} \dagger \widehat{z}(\boxed{\mathrm{Q}}\widehat{\delta}\ [z]\ \widehat{t}\,(\boxed{\mathrm{P}}\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle))$$

where $\mathbf{B}$ is a boolean, while $P$ and $Q$ are two arbitrary terms. Notice that, due to linearity, they must contain free names $\epsilon$ and $\delta$ respectively, otherwise the above term would not be linear.

**Remark** We have to assume that $\delta$ and $\epsilon$ are L-principal names of $P$ and $Q$, respectively. For simplicity we also assume that $P$ and $Q$ are closed, i.e., they do not have other free names. It is necessary to introduce these conditions which restrict the non-determinism in order to have the behavior we want.

The behavior we expect is the following; the computation should result in $P$ if we choose $\mathbf{B} = true$, otherwise it should result in $Q$ (if we take $\mathbf{B} = false$).

**Example 33** *On the one hand, for* $\mathbf{B} = true$*, the computation proceeds as follows:*

$Cond\{true,P,Q\} \triangleq$

$$(\widehat{y}\,(\widehat{x}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta)\,\widehat{\beta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{z}(Q\,\widehat{\delta}\ [z]\ \widehat{t}\,(P\,\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle))$$

$$
\begin{array}{rl}
\xrightarrow{ei\text{-}insert} & (Q\widehat{\delta} \dagger \widehat{y}(\widehat{x}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta))\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{act\text{-}R} & (Q\widehat{\delta}\nwarrow \widehat{y}(\widehat{x}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta))\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{\nwarrow\text{-}prop} & (\widehat{x}\,(Q\widehat{\delta} \nwarrow \widehat{y}(y \odot \langle x.\alpha\rangle))\,\widehat{\alpha}\cdot\beta)\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{\nwarrow\text{-}eras} & (\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\alpha}\cdot\beta)\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{ei\text{-}insert} & (P\widehat{\epsilon} \dagger \widehat{x}\langle x.\alpha\rangle)\widehat{\alpha} \dagger \widehat{u}\langle u.\sigma\rangle \\
\xrightarrow{ren\text{-}R} & (P\{\alpha/\epsilon\})\widehat{\alpha} \dagger \widehat{u}\langle u.\sigma\rangle \\
\xrightarrow{ren\text{-}R} & P\{\alpha/\epsilon\}\{\sigma/\alpha\} \ = \ P\{\sigma/\epsilon\}
\end{array}
$$

**Remark** Absent the condition $\delta$ is L-principal for $Q$, we could have decided to perform *act-L* in the second step (on the same cut). And if we choose $Q$ to be of the form $Q' \odot \delta$ we can construct a reduction sequence which results in $Q$. We may try to justify the introduction of those conditions by saying that it does not make much sense to plug in a component over an eraser.

**Example 34** *On the other hand, for* $\mathbf{B} = false$*, we have the following computation:*

$Cond\{false, P, Q\} \triangleq$

$$
\begin{array}{rl}
 & (\widehat{x}\,(\widehat{y}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta)\,\widehat{\beta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{z}(Q\,\widehat{\delta}\ [z]\ \widehat{t}\,(P\,\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle)) \\
\xrightarrow{ei\text{-}insert} & (Q\widehat{\delta} \dagger \widehat{x}(\widehat{y}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta))\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{act\text{-}R} & (Q\widehat{\delta}\nwarrow \widehat{x}(\widehat{y}\,(y \odot \langle x.\alpha\rangle)\,\widehat{\alpha}\cdot\beta))\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{\nwarrow\text{-}prop}{}^{+} & (\widehat{y}\,(y \odot (Q\widehat{\delta} \nwarrow \widehat{x}\langle x.\alpha\rangle))\,\widehat{\alpha}\cdot\beta)\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{\nwarrow\text{-}cap\text{-}deact} & (\widehat{y}\,(y \odot (Q\widehat{\delta} \dagger \widehat{x}\langle x.\alpha\rangle))\,\widehat{\alpha}\cdot\beta)\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{ren\text{-}R} & (\widehat{y}\,(y \odot Q\{\alpha/\delta\})\,\widehat{\alpha}\cdot\beta)\widehat{\beta} \dagger \widehat{t}(P\widehat{\epsilon}\ [t]\ \widehat{u}\,\langle u.\sigma\rangle) \\
\xrightarrow{ei\text{-}insert} & (P\widehat{\epsilon} \dagger \widehat{y}(y \odot Q\{\alpha/\delta\}))\widehat{\alpha} \dagger \widehat{u}\langle u.\sigma\rangle \\
\xrightarrow{act\text{-}R} & (P\widehat{\epsilon}\nwarrow \widehat{y}(y \odot Q\{\alpha/\delta\}))\widehat{\alpha} \dagger \widehat{u}\langle u.\sigma\rangle \\
\xrightarrow{\nwarrow\text{-}eras} & (Q\{\alpha/\delta\})\widehat{\alpha} \dagger \widehat{u}\langle u.\sigma\rangle \\
\xrightarrow{ren\text{-}R} & Q\{\alpha/\delta\}\{\sigma/\alpha\} \ = \ Q\{\sigma/\delta\}
\end{array}
$$

### 5.5.2 Natural numbers

A natural number is a term with two free innames and a single free outname. It is composed of capsules, importers and contractions.

**Definition 35** *We use* ${}_{x,y}\llbracket n \rrbracket_\alpha$ *to denote interpretation of a natural number* $n$ *in* ${}^*\mathcal{X}$*. The interpretation has free innames* $x, y$ *and a free outname* $\alpha$ *and it is defined as follows*

$$\begin{aligned} {}_{x,y}\llbracket 0 \rrbracket_\alpha &:= y \odot \langle x.\alpha \rangle \\ {}_{x,y}\llbracket n+1 \rrbracket_\alpha &:= y <^{\widehat{y_1}}_{\widehat{y_2}} \langle ({}_{x,y_1}\llbracket n \rrbracket_{\alpha_2})\, \widehat{\alpha_2}\ [y_2]\ \widehat{x_3}\, \langle x_3.\alpha \rangle], \quad n \geq 0 \end{aligned}$$

**Example 36** *We can derive*

$$\begin{aligned} {}_{x,y}\llbracket 1 \rrbracket_\alpha &\triangleq \langle x.\alpha_1 \rangle\, \widehat{\alpha_1}\ [y]\ \widehat{x_2}\, \langle x_2.\alpha \rangle \\ {}_{x,y}\llbracket 2 \rrbracket_\alpha &\triangleq y <^{\widehat{y_1}}_{\widehat{y_2}} \langle \underline{(\langle x.\alpha_1 \rangle\, \widehat{\alpha_1}\ [y_1]\ \widehat{x_2}\, \langle x_2.\alpha_2 \rangle)}\, \widehat{\alpha_2}\ [y_2]\ \widehat{x_3}\, \langle x_3.\alpha \rangle] \\ &\vdots \end{aligned}$$

**Remark** Another way to write ${}_{x,y}\llbracket 2 \rrbracket_\alpha$ is by changing the position of parenthesis[7]

$${}_{x,y}\llbracket 2 \rrbracket_\alpha \triangleq y <^{\widehat{y_1}}_{\widehat{y_2}} \langle \langle x.\alpha_1 \rangle\, \widehat{\alpha_1}\ [y_1]\ \widehat{x_2}\, \underline{(\langle x_2.\alpha_2 \rangle\, \widehat{\alpha_2}\ [y_2]\ \widehat{x_3}\, \langle x_3.\alpha \rangle)}],$$

but we will not go into detail here. More about equivalent terms will be said in Part III.

The basic operations are defined as follows:

$$\begin{aligned} {}_{x,y}\llbracket n+m \rrbracket_\alpha &:= y <^{\widehat{y_1}}_{\widehat{y_2}} \langle ({}_{x,y_1}\llbracket n \rrbracket_{\alpha_2}) \widehat{\alpha_2} \dagger \widehat{x_2}({}_{x_2,y_2}\llbracket m \rrbracket_\alpha)] \\ {}_{x,y}\llbracket n \times m \rrbracket_\alpha &:= (\widehat{x_1}\, ({}_{x_1,y}\llbracket n \rrbracket_{\alpha_2})\, \widehat{\alpha_2} \cdot \beta)\widehat{\beta} \dagger \widehat{y_1}({}_{x,y_1}\llbracket m \rrbracket_\alpha) \end{aligned}$$

The implementation of natural numbers will be revisited in the diagrammatic framework in Section 8.7 (page 119).

## 5.6 Extension of the ${}^*\mathcal{X}$-calculus

The ${}^*\mathcal{X}$ calculus has been presented as a counterpart of implicational fragment of classical logic. Its can be simply extended to cover other connectives. We extend the system to encompass negation, conjunction and disjunction. The same can be done in the diagrammatic setting (see Appendix 8.8).

[7] Underlining is used to make the comparison of terms easier.

## The syntax

The extension of the syntax is defined as follows:

$$
\begin{array}{lll}
P, Q ::= & \ldots & \\
| & x \,{}_{\neg}\, P\widehat{\alpha} & \textit{left-negation} \\
| & \widehat{x}P \,{}_{\neg}\, \alpha & \textit{right-negation} \\
| & x \,{}_{\vartriangle}\, \widehat{y}\widehat{z}P & \textit{left-conjunction} \\
| & P\widehat{\alpha}Q\widehat{\beta} \,{}_{\vartriangle}\, \gamma & \textit{right-conjunction} \\
| & x \,{}_{\triangledown}\, \widehat{y}P\widehat{z}Q & \textit{left-disjunction} \\
| & P\widehat{\alpha}\widehat{\beta} \,{}_{\triangledown}\, \gamma & \textit{right-disjunction}
\end{array}
$$

The notions of *linearity* and *free and bound names* naturally extend to this system. Principal names of new constructors (respectively $x, \alpha, x, \gamma, x, \gamma$) are considered as *L-principal*. This is natural since these terms code logical rules.

## The reduction rules

Since new constructors correspond to logical inference rules of sequent calculus, we extend the groups of logical actions. Also, the propagation group has to be extended to describe the cut-propagation for new constructors. Activation and deactivation groups remain the same with respect to extended notion of L-principal name.

**Logical rules** These rules define the cut-elimination for new constructors, when both cut names are L-introduced:

$$
\begin{array}{lll}
(\mathrm{not}_R\text{-}\mathrm{not}_L) & : (\widehat{y}P \,{}_{\neg}\, \alpha)\widehat{\alpha} \dagger \widehat{x}(x \,{}_{\neg}\, Q\widehat{\beta}) & \rightarrow Q\widehat{\beta} \dagger \widehat{y}P \\
(\mathrm{and}_R\text{-}\mathrm{and}_L) & : (P\widehat{\beta}Q\widehat{\gamma} \,{}_{\vartriangle}\, \alpha)\widehat{\alpha} \dagger \widehat{x}(x \,{}_{\vartriangle}\, \widehat{y}\widehat{z}R) & \rightarrow \begin{cases} P\widehat{\beta} \dagger \widehat{y}(Q\widehat{\gamma} \dagger \widehat{z}R) \\ Q\widehat{\gamma} \dagger \widehat{z}(P\widehat{\beta} \dagger \widehat{y}R) \end{cases} \\
(\mathrm{or}_R\text{-}\mathrm{or}_L) & : (P\widehat{\beta}\widehat{\gamma} \,{}_{\triangledown}\, \alpha)\widehat{\alpha} \dagger \widehat{x}(x \,{}_{\triangledown}\, \widehat{y}Q\widehat{z}R) & \rightarrow \begin{cases} (P\widehat{\beta} \dagger \widehat{y}Q)\widehat{\gamma} \dagger \widehat{z}R \\ (P\widehat{\gamma} \dagger \widehat{z}R)\widehat{\beta} \dagger \widehat{y}Q \end{cases}
\end{array}
$$

Notice that in the second and the third rule the conditions $y, z \in N(R)$ and $\beta, \gamma \in N(P)$ hold, respectively. For more intuition see the diagrammatic view given in the next chapter.

**Propagation rules** Propagation rules define cut-propagation through the structure of new constructors. We will introduce rules for both, left-propagation and right-propagation. Left-propagation is defined as:

$$
\begin{array}{lll}
(\text{not}_L \nearrow\text{-}prop) & : (x \neg P\widehat{\alpha})\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ x \neg (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \\
(\text{not}_R \nearrow\text{-}prop) & : (\widehat{x}P \neg \alpha)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ \widehat{x}(P\widehat{\beta} \nearrow \widehat{y}R) \neg \alpha,\ \ \alpha \neq \beta \\
 & & \\
(\text{and}_L \nearrow\text{-}prop) & : (x \vartriangle \widehat{z}\widehat{t}P)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ x \vartriangle \widehat{z}\widehat{t}(P\widehat{\beta} \nearrow \widehat{y}R) \\
(\text{and}_R \nearrow\text{-}prop_1) & : (P\widehat{\alpha}Q\widehat{\gamma} \vartriangle \delta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha}Q\widehat{\gamma} \vartriangle \delta,\ \ \delta \neq \beta,\ \beta \in N(P) \\
(\text{and}_R \nearrow\text{-}prop_2) & : (P\widehat{\alpha}Q\widehat{\gamma} \vartriangle \delta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ P\widehat{\alpha}(Q\widehat{\beta} \nearrow \widehat{y}R)\widehat{\gamma} \vartriangle \delta,\ \ \delta \neq \beta,\ \beta \in N(Q) \\
 & & \\
(\text{or}_L \nearrow\text{-}prop_1) & : (x \triangledown \widehat{z}P\widehat{t}Q)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ x \triangledown \widehat{z}(P\widehat{\beta} \nearrow \widehat{y}R)\widehat{t}Q,\ \ \beta \in N(P) \\
(\text{or}_L \nearrow\text{-}prop_2) & : (x \triangledown \widehat{z}P\widehat{t}Q)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ x \triangledown \widehat{z}P\widehat{t}(Q\widehat{\beta} \nearrow \widehat{y}R),\ \ \beta \in N(Q) \\
(\text{or}_R \nearrow\text{-}prop) & : (P\widehat{\alpha}\widehat{\gamma} \triangledown \delta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow\ (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha}\widehat{\gamma} \triangledown \delta,\ \ \delta \neq \beta
\end{array}
$$

Right-propagation is defined as:

$$
\begin{array}{lll}
(\nwarrow\text{not}_L\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(y \neg Q\widehat{\beta}) & \rightarrow\ y \neg (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta},\ \ x \neq y \\
(\nwarrow\text{not}_R\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(\widehat{y}Q \neg \beta) & \rightarrow\ \widehat{y}(P\widehat{\alpha} \nwarrow \widehat{x}Q) \neg \beta \\
 & & \\
(\nwarrow\text{and}_L\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(y \vartriangle \widehat{z}\widehat{t}Q) & \rightarrow\ y \vartriangle \widehat{z}\widehat{t}(P\widehat{\alpha} \nwarrow \widehat{x}Q),\ \ x \neq y \\
(\nwarrow\text{and}_R\text{-}prop_1) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta}R\widehat{\gamma} \vartriangle \delta) & \rightarrow\ (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta}R\widehat{\gamma} \vartriangle \delta,\ \ x \in N(Q) \\
(\nwarrow\text{and}_R\text{-}prop_2) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta}R\widehat{\gamma} \vartriangle \delta) & \rightarrow\ Q\widehat{\beta}(P\widehat{\alpha} \nwarrow \widehat{x}R)\widehat{\gamma} \vartriangle \delta,\ \ x \in N(R) \\
 & & \\
(\nwarrow\text{or}_L\text{-}prop_1) & : P\widehat{\alpha} \nwarrow \widehat{x}(y \triangledown \widehat{z}Q\widehat{t}R) & \rightarrow\ y \triangledown \widehat{z}(P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{t}R,\ \ x \neq y,\ x \in N(Q) \\
(\nwarrow\text{or}_L\text{-}prop_2) & : P\widehat{\alpha} \nwarrow \widehat{x}(y \triangledown \widehat{z}Q\widehat{t}R) & \rightarrow\ y \triangledown \widehat{z}P\widehat{t}(P\widehat{\alpha} \nwarrow \widehat{x}R),\ \ x \neq y,\ x \in N(R) \\
(\nwarrow\text{or}_R\text{-}prop) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\widehat{\beta}\widehat{\gamma} \triangledown \delta) & \rightarrow\ (P\widehat{\alpha} \nwarrow \widehat{x}Q)\widehat{\beta}\widehat{\gamma} \triangledown \delta
\end{array}
$$

## The type assignment

We define the type assignment for the other connectives by extending the type system defined in Figure 5.10, by the following:

$$\frac{P \vdots\ \Gamma \vdash \alpha : A, \Delta}{x\ {}_{\neg}\ P\widehat{\alpha} \vdots\ \Gamma, x : \neg A \vdash \Delta}\ (\neg\ L) \qquad \frac{P \vdots\ \Gamma, x : A \vdash \Delta}{\widehat{x}P\ {}_{\neg}\ \alpha \vdots\ \Gamma \vdash \alpha : \neg A, \Delta}\ (\neg\ R)$$

$$\frac{P \vdots\ \Gamma, y : A, z : B \vdash \Delta}{x\ {}_{\triangle}\ \widehat{y}\widehat{z}P \vdots\ \Gamma, x : A \wedge B \vdash \Delta}\ (\wedge L) \qquad \frac{P \vdots\ \Gamma \vdash \beta : A, \Delta \qquad Q \vdots\ \Gamma' \vdash \gamma : B, \Delta'}{P\widehat{\beta}Q\widehat{\gamma}\ {}_{\triangle}\ \alpha \vdots\ \Gamma, \Gamma' \vdash \alpha : A \wedge B, \Delta, \Delta'}\ (\wedge R)$$

$$\frac{P \vdots\ \Gamma, y : A \vdash \Delta \qquad Q \vdots\ \Gamma', z : B \vdash \Delta'}{x\ {}_{\triangledown}\ \widehat{y}Q\widehat{z}R \vdots\ \Gamma, \Gamma', x : A \vee B \vdash \Delta, \Delta'}\ (\vee L) \qquad \frac{P \vdots\ \Gamma \vdash \beta : A, \gamma : B, \Delta}{P\widehat{\beta}\widehat{\gamma}\ {}_{\triangledown}\ \alpha \vdots\ \Gamma \vdash \alpha : A \vee B, \Delta}\ (\vee R)$$

Computational content of a system in which some non-standard connectives were chosen as primitives (such as for example 'if and only if' and 'exclusive or') has been studied in [RS07], with the $\mathcal{X}$ calculus as an underlying language. An investigation in of various approaches of employing quantifiers in the $\mathcal{X}$ calculus has been studied in [SvB06].

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x_1.\delta\rangle :\cdot\ x_1 : A \vdash \delta : A}\,(ax) \quad \dfrac{}{\langle y.\beta\rangle :\cdot\ y : B \vdash \beta : B}\,(ax)}{\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle :\cdot\ x_1 : A, u : A \to B \vdash \beta : B}\,(\to L) \quad \dfrac{}{\langle z.\gamma\rangle :\cdot\ z : C \vdash \gamma : C}\,(ax)}{(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle)\,\widehat{\beta}\,[v]\,\widehat{z}\,\langle z.\gamma\rangle :\cdot\ v : B \to C, x_1 : A, u : B \to C \vdash \gamma : C}\,(\to L) \quad \dfrac{}{\langle x_2.\epsilon\rangle :\cdot\ x_2 : A \vdash \epsilon : A}\,(ax)}{\langle x_2.\epsilon\rangle\,\widehat{\epsilon}\,[w]\,\widehat{v}\,(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle)\,\widehat{\beta}\,[v]\,\widehat{z}\,\langle z.\gamma\rangle :\cdot\ x_1 : A, x_2 : A, u : A \to B, w : A \to B \to C \vdash \gamma : C}\,(\to L)}{x <^{\widehat{x_1}}_{\widehat{x_2}} \langle x_2.\epsilon\rangle\,\widehat{\epsilon}\,[w]\,\widehat{v}\,(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle)\,\widehat{\beta}\,[v]\,\widehat{z}\,\langle z.\gamma\rangle)] :\cdot\ x : A, u : A \to B, w : A \to B \to C \vdash \gamma : C}\,(cont\text{-}L)}{\widehat{x}\,(x <^{\widehat{x_1}}_{\widehat{x_2}} \langle x_2.\epsilon\rangle\,\widehat{\epsilon}\,[w]\,\widehat{v}\,(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle)\,\widehat{\beta}\,[v]\,\widehat{z}\,\langle z.\gamma\rangle)])\,\widehat{\gamma}\cdot\eta :\cdot\ u : A \to B, w : A \to B \to C \vdash \eta : A \to C}\,(\to R)}{\widehat{u}\,(\widehat{x}\,(x <^{\widehat{x_1}}_{\widehat{x_2}} \langle x_2.\epsilon\rangle\,\widehat{\epsilon}\,[w]\,\widehat{v}\,(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\,\widehat{y}\,\langle y.\beta\rangle)\,\widehat{\beta}\,[v]\,\widehat{z}\,\langle z.\gamma\rangle)])\,\widehat{\gamma}\cdot\eta)\,\widehat{\eta}\cdot\theta :\cdot\ w : A \to B \to C \vdash \theta : (A \to B) \to A \to C}\,(\to R)}{S :\cdot\ \vdash \alpha : (A \to B \to C) \to (A \to B) \to A \to C}\,(\to R)
$$

Figure 5.11: The proof tree for S-cominator

# Chapter 6

# The encoding of related calculi

The ${}^*\mathcal{X}$ calculus is a low-level language whose syntax is an extension of that of the $\mathcal{X}$ calculus, and therefore its reduction steps decompose reduction steps of $\mathcal{X}$, which on its own is also a low level language.

The expressive power of $\mathcal{X}$ has been illustrated in [vBLL05], by encoding various calculi, such as: $\lambda$, $\lambda x$ and $\lambda\mu$. Also the $\mathcal{X}$ calculus is encoded into $\lambda\mu$ in [AvB07]. The first hint on how to relate $\bar{\lambda}\mu\tilde{\mu}$ and Gentzen's sequent calculus for classical logic LK (which corresponds to $\mathcal{X}$) was already given by Curien and Herbelin in [CH00]. It was studied in detail through the $\lambda\xi$-calculus [Len03], where mutual embedings are presented. These results were used to give an elegant proof of strong normalization for the $\bar{\lambda}\mu\tilde{\mu}$-calculus.

Some works have considered the relation between $\mathcal{X}$ and the $\pi$-calculus. The $\pi$-calculus, [Mil95, SW01], is able to describe concurrent computations, including the communication between processes. The configurations of the interacting processes may change during the computation. The relation of $\mathcal{X}$ and $\pi$-calculus has been recently presented in [vBCV07], where the $\mathcal{X}$ calculus is encoded into $\pi$. This paper seeks for the intuition to what is computational meaning of cut-elimination from the point of view of $\pi$.

Some remarks aiming at essential points related to concurrency were given earlier by Urban [Urb00]. He suggested how a form of weak communication can be implemented, using quantifiers, into the classical sequent calculus. Besides that, it was noted that the approach where reduction is not seen as an equality preserving operation, is a standard approach in the calculi of concurrency. Take for example the non-deterministic choice opera-

tor, +, and the reduction: $P + Q \rightarrow P$ or $Q$. Moreover, the substitution mechanism in $\mathcal{X}$-like calculi in which only names may participate, is closer to the $\pi$-calculus than the substitution mechanism defined in the $\lambda$-calculus which involves terms.

Our view is that most of the features of the $\mathcal{X}$ calculus can also be shown for ${}^*\mathcal{X}$. Since the ${}^*\mathcal{X}$ calculus has a lower level of granularity, is expected to be at least as expressive as the $\mathcal{X}$ calculus. In case of potential implementation this model is better suited, since it introduces the possibility of controlling both duplication and erasure of parts of a program. In this chapter we study the relation between ${}^*\mathcal{X}$ and the following calculi; intuitionistic: $\lambda$, $\lambda\mathsf{x}$ and $\lambda\mathsf{lxr}$, and classical: $\mathcal{X}$ and $\bar{\lambda}\mu\tilde{\mu}$.

## 6.1 Relation between $\mathcal{X}$ calculus and ${}^*\mathcal{X}$ calculus

In this section we show the relation between $\mathcal{X}$-terms and ${}^*\mathcal{X}$-terms. We present the encodings in both directions, and study the relation between the computations. It is shown that $\mathcal{X}$-reduction steps are decomposed into more atomic steps of ${}^*\mathcal{X}$, due to the linearity and the presence of explicit terms for erasure and duplication. Finally, we study the relation between typing of $\mathcal{X}$-terms and typing of ${}^*\mathcal{X}$-terms.

### 6.1.1 From $\mathcal{X}$ to ${}^*\mathcal{X}$

We now describe how to encode $\mathcal{X}$-terms, which are possibly not linear, into terms of the ${}^*\mathcal{X}$ calculus. Before doing that we will introduce two operations to help us formulate the encoding. They will be used in the formal definition and their only purpose is to make definitions easier to read. The first operation, denoted by $\circledcirc$, adds *erasers* where needed.

**Definition 37 (Potential eraser: $\circledcirc$)** *The operation $\circledcirc$ is defined as follows:*

$$x \circledcirc P \circledcirc \alpha \;=\; \begin{cases} P, & x, \alpha \in N(P) \\ x \odot P, & x \notin N(P),\ \alpha \in N(P) \\ P \odot \alpha, & x \in N(P),\ \alpha \notin N(P) \\ x \odot P \odot \alpha, & x, \alpha \notin N(P) \end{cases}$$

The second operation, denoted by $\lhd(\ )\rhd$, adds *contractions* where needed. Typically this will happen when encoding terms which have two immediate subterms, denoted by $C\{P, Q\}$, and it will be used to prevent the multiple

occurrences of names. This operation also improves the readability of the encoding, although we could have used actual contractions.

**Definition 38 (Potential contractions: $\lhd(\ )\rhd$)** *The operation $\lhd(\ )\rhd$ is defined as follows:*

$$\mathcal{I}\lhd\Big(C\{P,Q\}\Big)\rhd\mathcal{O} \;=\; \begin{cases} C\{P,Q\}, & N(P)\cap N(Q)=\emptyset \\ \mathcal{I}<\frac{\widehat{\mathcal{I}_1}}{\widehat{\mathcal{I}_2}}\langle C\{P,Q\}\rangle\frac{\widehat{\mathcal{O}_1}}{\widehat{\mathcal{O}_2}}>\mathcal{O} & \text{when } N(P)\cap N(Q)\neq\emptyset, \\ & \text{where } \mathcal{I}=I(P)\cap I(Q) \\ & \qquad\ \mathcal{O}=O(P)\cap O(Q) \end{cases}$$

**Definition 39** *The encoding of $\mathcal{X}$-terms in ${}^*\mathcal{X}$ is defined by induction, as presented by Figure 6.1.*

$$\begin{array}{lcl}
\llbracket\langle x.\alpha\rangle\rrbracket^{*\mathcal{X}} & := & \langle x.\alpha\rangle \\[2ex]
\llbracket\widehat{x}\,P\,\widehat{\beta}\cdot\alpha\rrbracket^{*\mathcal{X}} & := & \Big(\widehat{x}\,(x\circledcirc\llbracket P\rrbracket^{*\mathcal{X}}\circledcirc\beta)\,\widehat{\beta}\cdot\alpha\Big)\rhd\alpha, \\[2ex]
\llbracket P\,\widehat{\alpha}\ [x]\ \widehat{y}\,Q\rrbracket^{*\mathcal{X}} & := & \mathcal{I}\lhd\Big((\llbracket P\rrbracket^{*\mathcal{X}}\circledcirc\alpha)\,\widehat{\alpha}\ [x]\ \widehat{y}\,(y\circledcirc\llbracket Q\rrbracket^{*\mathcal{X}})\Big)\rhd\mathcal{O}, \\
 & & \text{for } x\notin N(P), x\notin N(Q) \\[2ex]
\llbracket P\,\widehat{\alpha}\ [x]\ \widehat{y}\,Q\rrbracket^{*\mathcal{X}} & := & \mathcal{I}\lhd\Big(x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle(\llbracket P\{x_1/x\}\rrbracket^{*\mathcal{X}}\circledcirc\alpha)\,\widehat{\alpha}\ [x_2]\ \widehat{y}\,(y\circledcirc\llbracket Q\rrbracket^{*\mathcal{X}})]\Big)\rhd\mathcal{O}, \\
 & & \text{for } x\in N(P), x\notin N(Q) \\[2ex]
\llbracket P\,\widehat{\alpha}\ [x]\ \widehat{y}\,Q\rrbracket^{*\mathcal{X}} & := & \mathcal{I}\lhd\Big(x<\frac{\widehat{x_1}}{\widehat{x_2}}\langle(\llbracket P\rrbracket^{*\mathcal{X}}\circledcirc\alpha)\,\widehat{\alpha}\ [x_1]\ \widehat{y}\,(y\circledcirc\llbracket Q\{x_2/x\}\rrbracket^{*\mathcal{X}})]\Big)\rhd\mathcal{O}, \\
 & & \text{for } x\notin N(P), x\in N(Q) \\[2ex]
\llbracket P\,\widehat{\alpha}\ [x]\ \widehat{y}\,Q\rrbracket^{*\mathcal{X}} & := & \mathcal{I}\lhd\Big(x<\frac{\widehat{t}}{\widehat{x_3}}\langle t<\frac{\widehat{x_1}}{\widehat{x_2}}\langle(\llbracket P\{x_1/x\}\rrbracket^{*\mathcal{X}}\circledcirc\alpha)\,\widehat{\alpha}\ [x_2]\ \widehat{y}\,(y\circledcirc\llbracket Q\{x_3/x\}\rrbracket^{*\mathcal{X}})]]\Big)\rhd\mathcal{O}, \\
 & & \text{for } x\in N(P), x\in N(Q) \\[2ex]
\llbracket P\widehat{\alpha}\ \dagger\ \widehat{x}Q\rrbracket^{*\mathcal{X}} & := & \mathcal{I}\lhd\Big((\llbracket P\rrbracket^{*\mathcal{X}}\circledcirc\alpha)\widehat{\alpha}\ \dagger\ \widehat{x}(x\circledcirc\llbracket Q\rrbracket^{*\mathcal{X}})\Big)\rhd\mathcal{O},
\end{array}$$

Figure 6.1: Encoding the $\mathcal{X}$-terms into ${}^*\mathcal{X}$

Figure 6.1 defines the encoding of pure $\mathcal{X}$-terms in ${}^*\mathcal{X}$. Active cuts can be encoded in the following way:

$$[\![P\widehat{\alpha} \nearrow \widehat{x}Q]\!]^{*\mathcal{X}} := \mathcal{I} \lhd \Big( ([\![P]\!]^{*\mathcal{X}} \circledcirc \alpha)\widehat{\alpha} \nearrow \widehat{x}(x \circledcirc [\![Q]\!]^{*\mathcal{X}}) \Big) \rhd \mathcal{O}$$

$$[\![P\widehat{\alpha} \nwarrow \widehat{x}Q]\!]^{*\mathcal{X}} := \mathcal{I} \lhd \Big( ([\![P]\!]^{*\mathcal{X}} \circledcirc \alpha)\widehat{\alpha} \nwarrow \widehat{x}(x \circledcirc [\![Q]\!]^{*\mathcal{X}}) \Big) \rhd \mathcal{O}$$

**Remark** Notice that if the $\mathcal{X}$-term is linear, i.e., if there is no need to use the operations $\circledcirc$ and $\lhd(\ )\rhd$, we get simply

$$\begin{aligned}
[\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} &= \langle x.\alpha\rangle \\
[\![\widehat{x}\, P\, \widehat{\beta}\cdot\alpha]\!]^{*\mathcal{X}} &= \widehat{x}\, [\![P]\!]^{*\mathcal{X}}\, \widehat{\beta}\cdot\alpha \\
[\![P\, \widehat{\alpha}\ [x]\ \widehat{y}\, Q]\!]^{*\mathcal{X}} &= [\![P]\!]^{*\mathcal{X}}\, \widehat{\alpha}\ [x]\ \widehat{y}\, [\![Q]\!]^{*\mathcal{X}} \\
[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]^{*\mathcal{X}} &= [\![P]\!]^{*\mathcal{X}}\widehat{\alpha} \dagger \widehat{x}[\![Q]\!]^{*\mathcal{X}}
\end{aligned}$$

**Remark** The encoding is defined in such a way that none of the free names is lost. Notice that this is not the case with *occurrences* of free names. If a free name has multiple occurrences in $\mathcal{X}$-term, it will occur only once after the encoding.

**Lemma 40** *The encoding* $[\![\ ]\!]^{*\mathcal{X}}$ *preserves the set of free names.*

$$N(P) = N([\![P]\!]^{*\mathcal{X}})$$

**Proof:** By inspection of the encoding rules. □

**Example 41** *Take for example* $P = (\widehat{x}\,\langle x.\alpha\rangle\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha\rangle$*, where* $\alpha$ *as a free name occurs twice, and* $\widehat{\beta}$ *does not bind an occurrence of a free name. The encoding gives:*

$$[\![P]\!]^{*\mathcal{X}} = [(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}} > \alpha$$

*where* $\alpha$ *has only one occurrence, and* $\widehat{\beta}$ *does bind an occurrence of a free name.*

**Remark** We will sometimes annotate the arrow, $\rightarrow$, in order to ease the reading. We use $\xrightarrow{*\mathcal{X}}$ to denote ${}^*\mathcal{X}$-reduction and $\xrightarrow{\mathcal{X}}$ to denote $\mathcal{X}$-reduction.

**Simulation of $\mathcal{X}$-reduction** In what follows we show that the reduction rules of $\mathcal{X}$ can be simulated in ${}^*\mathcal{X}$. Initially we show that the notion of *introduced name* in $\mathcal{X}$ corresponds to the notion of *L-principal name* in ${}^*\mathcal{X}$.

**Lemma 42** *The notion of* introduced name *by a term in $\mathcal{X}$, and that of* L-principal name *of a term in ${}^*\mathcal{X}$, correspond to each other.*

1. *If $\alpha$ freshly introduced by $S$, then $\alpha$ is L-principal for $[\![S]\!]^{*\mathcal{X}}$*
2. *If $\alpha$ L-principal for $S$, then $\alpha$ is freshly introduced by $[\![S]\!]^{\mathcal{X}}$*

**Proof:**

1. Case: $S = \langle x.\alpha\rangle$. We have $[\![S]\!]^{*\mathcal{X}} = [\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} = \langle x.\alpha\rangle$, and thus $\alpha$ is L-principal for $\langle x.\alpha\rangle$.

   Case: $S = \widehat{x}\,P\,\widehat{\beta}\cdot\alpha$. Since $\alpha$ is freshly introduced $\alpha \notin N(P)$. We have $[\![S]\!]^{\mathcal{X}} = [\![\widehat{y}\,P\,\widehat{\beta}\cdot\alpha]\!]^{*\mathcal{X}} = \widehat{y}\,(y \circledcirc [\![P]\!]^{*\mathcal{X}} \circledcirc \beta)\,\widehat{\beta}\cdot\alpha$, and thus by definition $\alpha$ is L-principal for $[\![S]\!]^{*\mathcal{X}}$.

2. Case: $S = \langle x.\alpha\rangle$. We have $[\![S]\!]^{\mathcal{X}} = [\![\langle x.\alpha\rangle]\!]^{\mathcal{X}} = \langle x.\alpha\rangle$, where $\alpha$ is freshly introduced by $\langle x.\alpha\rangle$, by definition.

   Case: $S = \widehat{x}\,P\,\widehat{\beta}\cdot\alpha$. By linearity it stands that $\alpha \notin N(P)$. We have $S = [\![\widehat{y}\,P\,\widehat{\beta}\cdot\alpha]\!]^{\mathcal{X}} = \widehat{y}\,[\![P]\!]^{\mathcal{X}}\,\widehat{\beta}\cdot\alpha$, and thus $\alpha$ is freshly introduced by $[\![S]\!]^{\mathcal{X}}$.

It is not difficult to check that the same holds for innames. □

In the proof for simulation property we use slightly modified $\mathcal{X}$-propagation rules. The idea is to set the side conditions in a way that will prevent unnecessary duplication. The right-propagation rules of $\mathcal{X}$ are given in Figure 4.4 (page 27), whereas the modified version for left-propagation is given in Figure 6.2 (page 84). Modified right-propagation can be defined similarly. Remark that the last rule, namely ($\not\curvearrowright$-*prop-dupl*$_2$) assumes the side condition $Q \neq \langle x.\beta\rangle$.

**Theorem 43 (Simulation of $\mathcal{X}$-reduction)** *Let $P$ and $P'$ be $\mathcal{X}$-terms. Then the following holds:*

$$\textit{If } P \xrightarrow{\mathcal{X}} P' \quad \textit{then} \quad [\![P]\!]^{*\mathcal{X}} \xrightarrow{{}^*\mathcal{X}}{}^{+} (\mathcal{I}^{P\setminus P'}) \odot [\![P']\!]^{*\mathcal{X}} \odot (\mathcal{O}^{P\setminus P'})$$

**Proof:** The proof goes by inspecting the reduction rules and by induction on the structure of terms. We give the proof for some reduction rules.

$$
\begin{array}{lcl}
(\nearrow\text{-}eras) & : & \langle x.\alpha\rangle\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \langle x.\alpha\rangle, \quad \alpha \neq \beta \\
(\nearrow\text{-}deact) & : & \langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R \\
(\nearrow\text{-}prop) & : & (\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha, \quad \alpha \neq \beta \\
(\nearrow\text{-}prop\text{-}dupl\text{-}deact) & : & (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \begin{cases} (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R, & \beta \notin N(P) \\ (\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R, & \beta \in N(P) \end{cases} \\
(\nearrow\text{-}prop\text{-}dupl_1) & : & (P\,\widehat{\alpha}\;[x]\;\widehat{z}\,Q)\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \begin{cases} (P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\alpha}\;[x]\;\widehat{z}\,Q, & \\ \qquad\qquad \beta \in N(P),\ \beta \notin N(Q) & \\ P\,\widehat{\alpha}\;[x]\;\widehat{z}\,(Q\widehat{\beta} \nearrow \widehat{y}R), & \\ \qquad\qquad \beta \notin N(P),\ \beta \in N(Q) & \\ (P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\alpha}\;[x]\;\widehat{z}\,(Q\widehat{\beta} \nearrow \widehat{y}R), & \\ \qquad\qquad \beta \in N(P),\ \beta \in N(Q) & \end{cases} \\
(\nearrow\text{-}(c)\text{-}prop\text{-}deact) & : & (P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle)\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \begin{cases} P\widehat{\alpha} \dagger \widehat{y}R, & \beta \notin N(P) \\ (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{y}R, & \beta \in N(P) \end{cases} \\
(\nearrow\text{-}prop\text{-}dupl_2) & : & (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \nearrow \widehat{y}R \;\rightarrow\; \begin{cases} (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q, & \\ \qquad\qquad \beta \in N(P),\ \beta \notin N(Q) & \\ P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \nearrow \widehat{y}R), & \\ \qquad\qquad \beta \notin N(P),\ \beta \in N(Q) & \\ (P\widehat{\beta} \nearrow \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \nearrow \widehat{y}R), & \\ \qquad\qquad \beta \in N(P),\ \beta \in N(Q) & \end{cases}
\end{array}
$$

Figure 6.2: Modified left propagation in $\mathcal{X}$

**Logical rules:**

• Take the $(cap - ren)$ rule: $\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \rightarrow \langle y.\beta\rangle$. We have:

$$
\begin{array}{rcl}
[\![\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle]\!]^{*\mathcal{X}} & \triangleq & [\![\langle y.\alpha\rangle]\!]^{*\mathcal{X}}\widehat{\alpha} \dagger \widehat{x}[\![\langle x.\beta\rangle]\!]^{*\mathcal{X}} \\
& \triangleq & \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \\
& \rightarrow & \langle y.b\rangle \\
& \triangleq & [\![\langle y.\beta\rangle]\!]^{*\mathcal{X}}
\end{array}
$$

• Take the $(exp - ren)$ rule: $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle \rightarrow \widehat{y}\,P\,\widehat{\beta}\cdot\gamma,\ \alpha \notin N(P)$.
We have (assuming for simplicity that $\gamma \notin N(P)$):

$$
\begin{array}{rcl}
[\![(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle]\!]^{*\mathcal{X}} & \triangleq & [\![\widehat{y}\,P\,\widehat{\beta}\cdot\alpha]\!]^{*\mathcal{X}}\widehat{\alpha} \dagger \widehat{x}[\![\langle x.\gamma\rangle]\!]^{*\mathcal{X}} \\
& \triangleq & (\widehat{y}\,(y \circledcirc [\![P]\!]^{*\mathcal{X}} \circledcirc \beta)\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \dagger \widehat{x}\langle x.\gamma\rangle \\
& \rightarrow & \widehat{y}\,(y \circledcirc [\![P]\!]^{*\mathcal{X}} \circledcirc \beta)\,\widehat{\beta}\cdot\gamma \\
& \triangleq & [\![\widehat{y}\,P\,\widehat{\beta}\cdot\gamma]\!]^{*\mathcal{X}}
\end{array}
$$

• Take the $(imp-ren)$ rule: $\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R) \to Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R$, where $x \notin N(Q),\ x \notin N(R)$. We have:

$$\begin{aligned}
[\![\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R)]\!]^{*\mathcal{X}} &\triangleq [\![\langle y.\alpha\rangle]\!]^{*\mathcal{X}}\widehat{\alpha} \dagger \widehat{x}[\![Q\,\widehat{\beta}\,[x]\,\widehat{z}\,R]\!]^{*\mathcal{X}} \\
&\triangleq \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}(\mathcal{I} \lhd \Big(([\![Q]\!]^{*\mathcal{X}} \circledcirc \beta)\,\widehat{\beta}\,[x]\,\widehat{z}\,(z \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \rhd \mathcal{O}) \\
&\to \mathcal{I} \lhd \Big(([\![Q]\!]^{*\mathcal{X}} \circledcirc \beta)\,\widehat{\beta}\,[y]\,\widehat{z}\,(z \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \rhd \mathcal{O} \\
&\triangleq [\![Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R]\!]^{*\mathcal{X}}
\end{aligned}$$

For simplicity we assumed that $y \notin N(Q)$ and $y \notin N(R)$.

**Activation rules**

• Take the $(act-L)$ rule: $P\widehat{\alpha} \dagger \widehat{x}Q \to P\widehat{\alpha} \nearrow \widehat{x}Q$, if $\alpha$ not freshly introduced by $P$. We have:

$$\begin{aligned}
[\![P\widehat{\alpha} \dagger \widehat{x}Q]\!]^{*\mathcal{X}} &\triangleq \mathcal{I}^{P\cap Q} \lhd \Big(([\![P]\!]^{*\mathcal{X}} \circledcirc \alpha)\widehat{\alpha} \dagger \widehat{x}(x \circledcirc [\![Q]\!]^{*\mathcal{X}})\Big) \rhd \mathcal{O}^{P\cap Q} \\
&\xrightarrow{Lem.42} \mathcal{I}^{P\cap Q} \lhd \Big(([\![P]\!]^{*\mathcal{X}} \circledcirc \alpha)\widehat{\alpha} \nearrow \widehat{x}(x \circledcirc [\![Q]\!]^{*\mathcal{X}})\Big) \rhd \mathcal{O}^{P\cap Q} \\
&\triangleq [\![P\widehat{\alpha} \nearrow \widehat{x}Q]\!]^{*\mathcal{X}}
\end{aligned}$$

Similarly for the rule $(act-R)$.

**Propagation rules**

• Take the $(\nearrow - eras)$ rule: $\langle x.\alpha\rangle\widehat{\beta} \nearrow \widehat{y}R \to \langle x.\alpha\rangle$, where $\alpha \neq \beta$.
We will take into consideration the possibility that $x, \alpha \in N(R)$. Thus we have:

$$\begin{aligned}
[\![\langle x.\alpha\rangle\widehat{\beta} \nearrow \widehat{y}R]\!]^{*\mathcal{X}} &\triangleq x \lhd \Big(([\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} \odot \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \rhd \alpha \\
&\to x \lhd \Big(\mathcal{I}^R \odot [\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} \odot \mathcal{O}^R\Big) \rhd \alpha \\
&\dashrightarrow (\mathcal{I}^R \setminus x) \odot [\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} \odot (\mathcal{O}^R \setminus \alpha)
\end{aligned}$$

• Take the $(\nearrow - deact)$ rule: $\langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R \to \langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R$. We have:

$$\begin{aligned}
[\![\langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R]\!]^{*\mathcal{X}} &\triangleq x \lhd \Big([\![\langle x.\beta\rangle]\!]^{*\mathcal{X}}\widehat{\beta} \nearrow \widehat{y}(y \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \\
&\triangleq x \lhd \Big(\langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}(y \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \\
&\to x \lhd \Big(\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}(y \circledcirc [\![R]\!]^{*\mathcal{X}})\Big) \\
&\triangleq [\![\langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R]\!]^{*\mathcal{X}}
\end{aligned}$$

• Take the $(\nearrow - prop)$ rule: $(\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R \to \widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha$, $\alpha \neq \beta$.
We assume for simplicity $N(\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha) \cap N(R) = \emptyset$. We have:

$$\begin{aligned}
\llbracket(\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}R\rrbracket^{*\mathcal{X}} &\triangleq (\llbracket\widehat{x}\,P\,\widehat{\gamma}\cdot\alpha\rrbracket^{*\mathcal{X}} \circledcirc \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\triangleq ((\widehat{x}\,(x \circledcirc \llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \gamma)\,\widehat{\gamma}\cdot\alpha) \circledcirc \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\equiv (\widehat{x}\,((x \circledcirc \llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \gamma) \circledcirc \beta)\,\widehat{\gamma}\cdot\alpha)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\to \widehat{x}\,(((x \circledcirc \llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \gamma) \circledcirc \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}))\,\widehat{\gamma}\cdot\alpha \\
&\equiv \widehat{x}\,(x \circledcirc ((\llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})) \circledcirc \gamma)\,\widehat{\gamma}\cdot\alpha \\
&\triangleq \widehat{x}\,(x \circledcirc (P\widehat{\beta} \nearrow \widehat{y}R) \circledcirc \gamma)\,\widehat{\gamma}\cdot\alpha \\
&\triangleq \llbracket\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\alpha\rrbracket^{*\mathcal{X}}
\end{aligned}$$

• Take the first case of (*↗-prop-dupl-deact*) rule:
$(\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R \;\to\; (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R$, where $\beta \notin N(P)$. We assume $N(P) \cap N(R) = \emptyset$, then we have:

$$\begin{aligned}
\llbracket(\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R\rrbracket^{*\mathcal{X}} &\triangleq \llbracket\widehat{x}\,P\,\widehat{\gamma}\cdot\beta\rrbracket^{*\mathcal{X}}\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\triangleq (\widehat{x}\,(x \circledcirc \llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \gamma)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\to (\widehat{x}\,(x \circledcirc \llbracket P\rrbracket^{*\mathcal{X}} \circledcirc \gamma)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\triangleq \llbracket(\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R\rrbracket^{*\mathcal{X}}
\end{aligned}$$

• Take the second case of (*↗-prop-dupl-deact*) rule:
$(\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R \;\to\; (\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R$, where $\beta \in N(P)$. We assume for simplicity $N(P) \cap N(R) = \emptyset$, then we have:

$$\begin{aligned}
&\llbracket(\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R\rrbracket^{*\mathcal{X}} \triangleq \llbracket\widehat{x}\,P\,\widehat{\gamma}\cdot\beta\rrbracket^{*\mathcal{X}}\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\triangleq ([\underline{\widehat{x}\,(x \odot (\llbracket P\rrbracket^{*\mathcal{X}}\{\beta_1/\beta\}) \odot \gamma)\,\widehat{\gamma}\cdot\beta_2}\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}} > \beta)\widehat{\beta} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}) \\
&\to \mathcal{I}^R \lhd \Big(((\underline{\widehat{x}\,(x \odot (\llbracket P\rrbracket^{*\mathcal{X}}\{\beta_1/\beta\}) \odot \gamma)\,\widehat{\gamma}\cdot\beta_2})\widehat{\beta_1} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}}))\widehat{\beta_2} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})\Big) \rhd \mathcal{O}^R \\
&\to \mathcal{I}^R \lhd \Big((\widehat{x}\,(x \circledcirc (\llbracket P\rrbracket^{*\mathcal{X}}\{\beta_1/\beta\}\widehat{\beta_1} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})) \circledcirc \gamma)\,\widehat{\gamma}\cdot\beta_2)\widehat{\beta_2} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})\Big) \rhd \mathcal{O}^R \\
&\to \mathcal{I}^R \lhd \Big((\widehat{x}\,(x \circledcirc (\llbracket P\rrbracket^{*\mathcal{X}}\{\beta_1/\beta\}\widehat{\beta_1} \nearrow \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})) \circledcirc \gamma)\,\widehat{\gamma}\cdot\beta_2)\widehat{\beta_2} \dagger \widehat{y}(y \circledcirc \llbracket R\rrbracket^{*\mathcal{X}})\Big) \rhd \mathcal{O}^R \\
&\triangleq \llbracket(\widehat{x}\,(P\widehat{\beta} \nearrow \widehat{y}R)\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R\rrbracket^{*\mathcal{X}}
\end{aligned}$$

The proof goes similarly for other propagation rules. □

**Preservation of types** We now show that the encoding preserves types. In the typed $\mathcal{X}$ calculus contexts $\Gamma$ and $\Delta$ may contain some auxiliary pairs (name,type). This is due to the fact that weakening is implicit in $\mathcal{X}$, i.e., it is not controlled explicitly. We have to keep that in mind when formulating the lemma.

**Lemma 44 (Preservation of types)** *If $P$ is an arbitrary $\mathcal{X}$-term such that $P :\cdot\ \Gamma \vdash \Delta$, then*

$$((dom(\Gamma)) \setminus I(P)) \odot [\![P]\!]^{*\mathcal{X}} \odot ((dom(\Delta)) \setminus O(P)) :\cdot\ \Gamma \vdash \Delta$$

**Proof:** The proof works by case analysis and induction on the structure of terms. We give the detail for encoding of capsule and exporter, whereas the other cases work the same way.

• Rule: $[\![\langle x.\alpha\rangle]\!]^{*\mathcal{X}} := \langle x.\alpha\rangle$.
If $\langle x.\alpha\rangle :\cdot\ \Gamma \vdash \Delta$ where $x : A \in \Gamma$ and $\alpha : A \in \Delta$, then, in ${}^*\mathcal{X}$ we have: $\langle x.\alpha\rangle :\cdot\ x : A \vdash \alpha : A$, which is equivalent to:

$$(dom(\Gamma) \setminus x) \odot \langle x.\alpha\rangle \odot (dom(\Delta) \setminus \alpha) :\cdot\ \Gamma \vdash \Delta$$

• Rule: $[\![\widehat{x}\, P\, \widehat{\beta} \cdot \alpha]\!]^{*\mathcal{X}} := \left(\widehat{x}\,(x \circledcirc [\![P]\!]^{*\mathcal{X}} \circledcirc \beta)\, \widehat{\beta} \cdot \alpha\right) \rhd \alpha$.
If we assume the most generic case, namely for $x, \beta \notin N(P)$ and $\alpha \in N(P)$, then the encoding gives:

$$[\![\widehat{x}\, P\, \widehat{\beta} \cdot \alpha]\!]^{*\mathcal{X}} := [\widehat{x}\,(x \odot ([\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\}) \odot \beta)\, \widehat{\beta} \cdot \alpha_1\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}} > \alpha$$

On the one hand we have:

$$\frac{P :\cdot\ \Gamma \vdash \alpha : A \to B, \Delta}{\widehat{x}\, P\, \widehat{\beta} \cdot \alpha :\cdot\ \Gamma \vdash \alpha : A \to B, \Delta}\,(\to R)$$

where, as stated previously, $x : A \in \Gamma,\ \beta : B \in \Delta$.

On the other hand,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\![P]\!]^{*\mathcal{X}} :\cdot\ \Gamma \vdash \alpha : A \to B, \Delta}{[\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\} :\cdot\ \Gamma \vdash \alpha_1 : A \to B, \Delta}\,(ren)}{x \odot [\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\} :\cdot\ \Gamma, x : A \vdash \alpha_1 : A \to B, \Delta}\,(weak\text{-}L)}{x \odot [\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\} \odot \beta :\cdot\ \Gamma, x : A \vdash \alpha_1 : A \to B, \beta : B, \Delta}\,(weak\text{-}R)}{\widehat{x}\,(x \odot [\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\} \odot \beta)\, \widehat{\beta} \cdot \alpha :\cdot\ \Gamma \vdash \alpha_1 : A \to B, \alpha_2 : A \to B, \Delta}\,(\to R)}{[\widehat{x}\,(x \odot [\![P]\!]^{*\mathcal{X}}\{\alpha_1/\alpha\} \odot \beta)\, \widehat{\beta} \cdot \alpha\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}} > \alpha :\cdot\ \Gamma \vdash \alpha : A \to B, \Delta}\,(cont\text{-}R)$$

□

### 6.1.2 From ${}^*\mathcal{X}$ to $\mathcal{X}$

Now we investigate the opposite direction. We show how to represent ${}^*\mathcal{X}$-terms by $\mathcal{X}$-terms and then we show how ${}^*\mathcal{X}$-reductions are simulated by $\mathcal{X}$-reductions.

**Definition 45 (Encoding ${}^*\mathcal{X}$ into $\mathcal{X}$)** *The encoding of ${}^*\mathcal{X}$-terms in $\mathcal{X}$ calculus is defined inductively as shown by Figure* 6.2.

$$
\begin{array}{rcl}
\llbracket\langle x.\alpha\rangle\rrbracket^{\mathcal{X}} & := & \langle x.\alpha\rangle \\
\llbracket\widehat{x}\,P\,\widehat{\beta}\cdot\alpha\rrbracket^{\mathcal{X}} & := & \widehat{x}\,\llbracket P\rrbracket^{\mathcal{X}}\,\widehat{\beta}\cdot\alpha \\
\llbracket P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}}\,\widehat{\alpha}\;[x]\;\widehat{y}\,\llbracket Q\rrbracket^{\mathcal{X}} \\
\llbracket P\widehat{\alpha}\;\dagger\;\widehat{x}Q\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}}\widehat{\alpha}\;\dagger\;\widehat{x}\llbracket Q\rrbracket^{\mathcal{X}} \\
\llbracket x<\frac{\widehat{y}}{\widehat{z}}\langle P]\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}}\{x/y\}\{x/z\} \\
\llbracket[P\rangle\frac{\widehat{\beta}}{\widehat{\gamma}}>\alpha\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}}\{\alpha/\beta\}\{\alpha/\gamma\} \\
\llbracket x\odot P\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}} \\
\llbracket P\odot\alpha\rrbracket^{\mathcal{X}} & := & \llbracket P\rrbracket^{\mathcal{X}}
\end{array}
$$

Figure 6.3: Encoding the ${}^*\mathcal{X}$-terms into $\mathcal{X}$

Encodings are defined without considering the active cuts but it is not difficult to extend it:

$$\llbracket P\widehat{\alpha}\nearrow\widehat{x}Q\rrbracket^{\mathcal{X}} := \llbracket P\rrbracket^{\mathcal{X}}\widehat{\alpha}\nearrow\widehat{x}\llbracket Q\rrbracket^{\mathcal{X}}$$

$$\llbracket P\widehat{\alpha}\nwarrow\widehat{x}Q\rrbracket^{\mathcal{X}} := \llbracket P\rrbracket^{\mathcal{X}}\widehat{\alpha}\nwarrow\widehat{x}\llbracket Q\rrbracket^{\mathcal{X}}$$

The encoding $\llbracket\ \rrbracket^{\mathcal{X}}$ does the opposite to $\llbracket\ \rrbracket^{{}^*\mathcal{X}}$. Namely, it simply removes erasers and duplicators from terms (some renamings are also performed). That is the reason for a possible decrease of free names after the encoding.

**Lemma 46** *The encoding $\llbracket\ \rrbracket^{\mathcal{X}}$ preserves or decreases the set of free names.*

$$N(P)\subseteq N(\llbracket P\rrbracket^{\mathcal{X}})$$

**Proof:** This can be checked by analyzing the encoding rules. The set of free names is preserved by most of the encoding rules, and decreased when encoding erasers. □

## 6.2 Relation with the $\bar{\lambda}\mu\tilde{\mu}$-calculus

In this section we present the encoding of $\bar{\lambda}\mu\tilde{\mu}$-calculus into ${}^*\mathcal{X}$ calculus.

Terms ($v$) :

$$\begin{aligned}
[\![x]\!]^{*\mathcal{X}}_{\alpha} &:= \langle x.\alpha \rangle \\
[\![\lambda x.v]\!]^{*\mathcal{X}}_{\alpha} &:= \widehat{x}\,(x \circledcirc [\![v]\!]^{*\mathcal{X}}_{\beta})\,\widehat{\beta}\cdot\alpha \\
[\![\mu\beta.c]\!]^{*\mathcal{X}}_{\alpha} &:= ([\![c]\!]^{*\mathcal{X}} \circledcirc \beta)\{\alpha/\beta\}
\end{aligned}$$

Contexts ($e$) :

$$\begin{aligned}
[\![\alpha]\!]^{*\mathcal{X}}_{x} &:= \langle x.\alpha \rangle \\
[\![v \cdot e]\!]^{*\mathcal{X}}_{x} &:= \mathcal{I} \lhd \left( [\![v]\!]^{*\mathcal{X}}_{\alpha}\,\widehat{\alpha}\ [x]\ \widehat{y}\,[\![e]\!]^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \\
[\![\tilde{\mu}y.c]\!]^{*\mathcal{X}}_{x} &:= (y \circledcirc [\![c]\!]^{*\mathcal{X}})\{x/y\}
\end{aligned}$$

Commands ($c$) :

$$[\![\langle v \parallel e \rangle]\!]^{*\mathcal{X}} := \mathcal{I} \lhd \left( [\![v]\!]^{*\mathcal{X}}_{\alpha}\widehat{\alpha} \dagger \widehat{x}[\![e]\!]^{*\mathcal{X}}_{x} \right) \rhd \mathcal{O}$$

Figure 6.4: Encoding $\bar{\lambda}\mu\tilde{\mu}$-calculus into ${}^*\mathcal{X}$

**Remark** Notice that $[\![\ ]\!]^{*\mathcal{X}}_{\alpha}$ is used for terms, $[\![\ ]\!]^{*\mathcal{X}}_{x}$ is used for contexts, and $[\![\ ]\!]^{*\mathcal{X}}$ - for commands.

**Remark** Meta substitution of $\bar{\lambda}\mu\tilde{\mu}$-calculus is represented in the following way:

$$\begin{aligned}
[\![c[e/\alpha]]\!]^{*\mathcal{X}} &:= ([\![c]\!]^{*\mathcal{X}} \circledcirc \alpha)\widehat{\alpha} \nearrow \widehat{x}[\![e]\!]^{*\mathcal{X}}_{x} \\
[\![c[v/x]]\!]^{*\mathcal{X}} &:= [\![v]\!]^{*\mathcal{X}}_{\alpha}\widehat{\alpha} \nwarrow \widehat{x}(x \circledcirc [\![c]\!]^{*\mathcal{X}})
\end{aligned}$$

This encoding relies on that presented in [Len03] and [vBLL05], where the $\bar{\lambda}\mu\tilde{\mu}$-calculus was studied in relation with $\lambda\xi$ and $\mathcal{X}$, respectively. The encoding is given in Figure 6.4. The operations $\circledcirc$ and $\lhd(\ )\rhd$ are given in Definitions 37 and 38, respectively.

**Lemma 47 (Simulation of the $\bar{\lambda}\mu\tilde{\mu}$-reduction)**

$$\textit{If}\ \ P \xrightarrow{\bar{\lambda}\mu\tilde{\mu}} P' \quad \textit{then} \quad [\![P]\!]^{*\mathcal{X}} \xrightarrow{{}^*\mathcal{X}}{}^{+} [\![P']\!]^{*\mathcal{X}}$$

**Proof:** By case analysis and induction on the structure of terms.

• Consider the first reduction rule: $\langle \lambda x.v_1 \parallel v_2 \cdot e\rangle \rightarrow \langle v_2 \parallel \tilde{\mu}x.\langle v_1 \parallel e\rangle\rangle$.
– On the one hand, the encoding of the left-hand side gives:

$$
\begin{array}{ll}
& [\![\langle \lambda x.v_1 \parallel v_2 \cdot e\rangle]\!]^{*\mathcal{X}} \triangleq \\
\triangleq & \mathcal{I}' \lhd \Big( [\![\lambda x.v_1]\!]^{*\mathcal{X}}_{\beta} \widehat{\beta} \dagger \widehat{y} [\![v_2 \cdot e]\!]^{*\mathcal{X}}_{y} \Big) \rhd \mathcal{O}' \\
\triangleq & \mathcal{I}' \lhd \Big( (\widehat{x}\,(x \circledcirc [\![v_1]\!]^{*\mathcal{X}}_{\gamma})\,\widehat{\gamma} \cdot \beta)\widehat{\beta} \dagger \widehat{y}(\mathcal{I}'' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta}\ [y]\ \widehat{z}\, [\![e]\!]^{*\mathcal{X}}_{z} \Big) \rhd \mathcal{O}'') \Big) \rhd \mathcal{O}'
\end{array}
$$

where the possible contractions are with $\mathcal{I}' = \mathcal{I}^{[\![v_1]\!]^{*\mathcal{X}}_{\gamma} \cap [\![v_2]\!]^{*\mathcal{X}}_{\delta} \cap [\![e]\!]^{*\mathcal{X}}_{z}}$ and $\mathcal{I}'' = \mathcal{I}^{[\![v_2]\!]^{*\mathcal{X}}_{\delta} \cap [\![e]\!]^{*\mathcal{X}}_{z}}$ (notice that these lists can also be empty). Similar definition is for outnames. By the condition of linearity $y \notin \mathcal{I}''$. If $\mathcal{I}''$,$\mathcal{O}''$ are not empty, we have to perform right activation followed by propagations and deactivation, in order to apply *ei-insert* rule:

$$
\begin{array}{ll}
\xrightarrow{act-R} & \mathcal{I}' \lhd \Big( (\widehat{x}\,(x \circledcirc [\![v_1]\!]^{*\mathcal{X}}_{\gamma})\,\widehat{\gamma} \cdot \beta)\widehat{\beta} \nwarrow \widehat{y}(\mathcal{I}'' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta}\ [y]\ \widehat{z}\, [\![e]\!]^{*\mathcal{X}}_{z} \Big) \rhd \mathcal{O}'') \Big) \rhd \mathcal{O}' \\
\xrightarrow{\nwarrow -prop}{}^{+} & \mathcal{I}' \lhd \Big( \mathcal{I}'' \lhd \Big( (\widehat{x}\,((x \circledcirc [\![v_1]\!]^{*\mathcal{X}}_{\gamma}))\,\widehat{\gamma} \cdot \beta)\widehat{\beta} \nwarrow \widehat{y}([\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta}\ [y]\ \widehat{z}\, [\![e]\!]^{*\mathcal{X}}_{z}) \Big) \rhd \mathcal{O}'' \Big) \rhd \mathcal{O}' \\
\xrightarrow{\nwarrow -deact_2} & \mathcal{I}' \lhd \Big( \mathcal{I}'' \lhd \Big( (\widehat{x}\,((x \circledcirc [\![v_1]\!]^{*\mathcal{X}}_{\gamma}))\,\widehat{\gamma} \cdot \beta)\widehat{\beta} \dagger \widehat{y}([\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta}\ [y]\ \widehat{z}\, [\![e]\!]^{*\mathcal{X}}_{z}) \Big) \rhd \mathcal{O}'' \Big) \rhd \mathcal{O}' \\
\xrightarrow{ei-insert} & \mathcal{I}' \lhd \Big( \mathcal{I}'' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{x}((x \circledcirc [\![v_1]\!]^{*\mathcal{X}}_{\gamma})\widehat{\gamma} \dagger \widehat{z} [\![e]\!]^{*\mathcal{X}}_{z}) \Big) \rhd \mathcal{O}'' \Big) \rhd \mathcal{O}'
\end{array}
$$

– On the other hand we have:

$$
\begin{array}{ll}
& [\![\langle v_2 \parallel \tilde{\mu}x.\langle v_1 \parallel e\rangle\rangle]\!]^{*\mathcal{X}} \triangleq \\
\triangleq & \mathcal{I}' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{y} [\![\tilde{\mu}x.\langle v_1 \parallel e\rangle]\!]^{*\mathcal{X}}_{y} \Big) \rhd \mathcal{O}' \\
\triangleq & \mathcal{I}' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{y}((x \circledcirc [\![\langle v_1 \parallel e\rangle]\!]^{*\mathcal{X}})\{y/x\}) \Big) \rhd \mathcal{O}' \\
= & \mathcal{I}' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{y}(y \circledcirc [\![\langle v_1 \parallel e\rangle]\!]^{*\mathcal{X}}) \Big) \rhd \mathcal{O}' \\
= & \mathcal{I}' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{x}(x \circledcirc [\![\langle v_1 \parallel e\rangle]\!]^{*\mathcal{X}}) \Big) \rhd \mathcal{O}' \\
\triangleq & \mathcal{I}' \lhd \Big( [\![v_2]\!]^{*\mathcal{X}}_{\delta} \widehat{\delta} \dagger \widehat{x}(x \circledcirc (\mathcal{I}''' \lhd \Big( [\![v_1]\!]^{*\mathcal{X}}_{\gamma} \widehat{\gamma} \dagger \widehat{z} [\![e]\!]^{*\mathcal{X}}_{z} \Big) \rhd \mathcal{O}''')) \Big) \rhd \mathcal{O}'
\end{array}
$$

where $\mathcal{I}'$ has already been defined, and $\mathcal{I}''' = \mathcal{I}^{[\![v_1]\!]^{*\mathcal{X}}_{\gamma} \cap [\![e]\!]^{*\mathcal{X}}_{z}}$. Due to liearity we have $x \notin \mathcal{I}'''$, and therefore we may consider the variant of a term when contraction are pulled at the top level. Thus we obtain the two resulting terms which are equivalent modulo congruence rules:

$$
z{<}\frac{\widehat{y}}{\widehat{x_3}}\langle y{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P]] \equiv z{<}\frac{\widehat{x_1}}{\widehat{y}}\langle y{<}\frac{\widehat{x_2}}{\widehat{x_3}}\langle P]] \quad \text{and} \quad [[P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}{>}\beta\rangle\frac{\widehat{\beta}}{\widehat{\alpha_3}}{>}\gamma \equiv [[P\rangle\frac{\widehat{\alpha_2}}{\widehat{\alpha_3}}{>}\beta\rangle\frac{\widehat{\alpha_1}}{\widehat{\beta}}{>}\gamma
$$

• Consider the second reduction rule: $\langle \mu\alpha.c \parallel e\rangle \;\rightarrow\; c[e/\alpha]$.
– On the one hand, the encoding of the left-hand side gives:

$$\begin{array}{rl} & \llbracket\langle \mu\alpha.c \parallel e\rangle\rrbracket^{*\mathcal{X}} \triangleq \\ \triangleq & \mathcal{I} \lhd \left( \llbracket \mu\alpha.c \rrbracket^{*\mathcal{X}}_{\beta} \widehat{\beta} \dagger \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \\ \triangleq & \mathcal{I} \lhd \left( (\llbracket c \rrbracket^{*\mathcal{X}} \circledcirc \alpha\{\beta/\alpha\}) \widehat{\beta} \dagger \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \\ = & \mathcal{I} \lhd \left( (\llbracket c \rrbracket^{*\mathcal{X}} \circledcirc \beta) \widehat{\beta} \dagger \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \\ \xrightarrow{act-R} & \mathcal{I} \lhd \left( (\llbracket c \rrbracket^{*\mathcal{X}} \circledcirc \beta) \widehat{\beta} \nwarrow \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \\ = & \mathcal{I} \lhd \left( (\llbracket c \rrbracket^{*\mathcal{X}} \circledcirc \alpha) \widehat{\alpha} \nwarrow \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \end{array}$$

– On the other hand:

$$\begin{array}{rl} & \llbracket\langle \mu\alpha.c \parallel e\rangle\rrbracket^{*\mathcal{X}} \triangleq \\ \triangleq & \mathcal{I} \lhd \left( (\llbracket c \rrbracket^{*\mathcal{X}} \circledcirc \alpha) \widehat{\alpha} \nwarrow \widehat{y} \llbracket e \rrbracket^{*\mathcal{X}}_{y} \right) \rhd \mathcal{O} \end{array}$$

where $\mathcal{I} = \mathcal{I}^{\llbracket c \rrbracket^{*\mathcal{X}} \cap \llbracket e \rrbracket^{*\mathcal{X}}_{y}}$, and similarly for outnames.

• The proof goes similarly for the third rule: $\langle v \parallel \tilde{\mu}x.c\rangle \;\rightarrow\; c[v/x]$.

□

## 6.3 Encoding the intuitionistic calculi

In this section we define the encoding of the $\lambda$-calculus in ${}^*\mathcal{X}$. In addition we also define the encodings of $\lambda$x-calculus and $\lambda$lxr-calculus.

The results we present in this section rely on the encodings of various calculi in the $\mathcal{X}$-calculus, as presented in [vBLL05].

### 6.3.1 Encoding the $\lambda$-calculus

Before giving the encoding, we define an operation $\circledcirc$ which adds erasers where needed. This will simplify the definition of encoding.

$$x \circledcirc \llbracket M \rrbracket_{\alpha} = \begin{cases} x \odot \llbracket M \rrbracket_{\alpha}, & x \notin fv(M) \\ \llbracket M \rrbracket_{\alpha}, & x \in fv(M) \end{cases}$$

Notice that $P$ is a $\lambda$-calculus term, and that $x$ is here a variable of the $\lambda$-calculus (hence the use $fv$ for "free variables"). The same letters are used for variables as for innames in ${}^*\mathcal{X}$. Moreover, they correspond to each other

in the sense that a free variable in $\lambda$-calculus becomes a free inname in ${}^*\mathcal{X}$ after the encoding.

The encoding of $\lambda$-calculus in ${}^*\mathcal{X}$ calculus is defined by induction as presented:

$$
\begin{array}{rcl}
\llbracket x \rrbracket_\alpha & := & \langle x.\alpha \rangle \\
\llbracket \lambda x.M \rrbracket_\alpha & := & \widehat{x}\,(x \circledcirc \llbracket M \rrbracket_\beta)\,\widehat{\beta}\cdot\alpha \\
\llbracket MN \rrbracket_\alpha & := & \mathcal{I}<\widehat{\frac{\mathcal{I}_1}{\mathcal{I}_2}}\langle \llbracket M_1 \rrbracket_\gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket N_2 \rrbracket_\beta\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle)\rangle, \\
 & & \text{where:} \\
 & & \mathcal{I} = fv(M) \cap fv(N) \\
 & & \llbracket M_1 \rrbracket_\gamma = ind(\llbracket M \rrbracket_\gamma, \mathcal{I}, 1) \\
 & & \llbracket N_2 \rrbracket_\beta = ind(\llbracket N \rrbracket_\beta, \mathcal{I}, 2)
\end{array}
$$

Figure 6.5: Encoding the $\lambda$-calculus

Notice that the encoding $\llbracket\ \rrbracket_\alpha$ preserves the set of free variables, which appear as free innames after the encoding. Since free variables correspond to free innames, we can conclude that $\lambda$-terms are represented by ${}^*\mathcal{X}$-terms with zero or more free innames and one free outname (obtained by construction as the result of encoding).

**Example 48** *We wish to illustrate how the general definition works, and therefore we show the encoding of several simple $\lambda$-terms:*

$$
\begin{array}{rcl}
\llbracket \lambda x.M \rrbracket_\alpha & \triangleq & \widehat{x}\,\llbracket M \rrbracket_\beta\,\widehat{\beta}\cdot\alpha, \\
 & & \textit{when } x \in fv(M) \\
\llbracket MN \rrbracket_\alpha & \triangleq & \llbracket M \rrbracket_\gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket N \rrbracket_\beta\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle), \\
 & & \textit{when } fv(M) \cap fv(N) = \emptyset \\
\llbracket MN \rrbracket_\alpha & \triangleq & z<\widehat{\frac{z_1}{z_2}}\langle \llbracket M_1 \rrbracket_\gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket N_2 \rrbracket_\beta\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle)\rangle, \\
 & & \textit{when } fv(M) \cap fv(N) = z \\
 & & \textit{where } M_1 = ind(M, z, 1) \textit{ and } N_2 = ind(N, z, 2)
\end{array}
$$

$$
\begin{array}{rcl}
\llbracket zz \rrbracket_\alpha & \triangleq & z<\widehat{\frac{z_1}{z_2}}\langle \llbracket z \rrbracket_\gamma\{z_1/z\}\widehat{\gamma} \dagger \widehat{x}\llbracket z \rrbracket_\beta\{z_2/z\}\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle] \\
 & \triangleq & z<\widehat{\frac{z_1}{z_2}}\langle \llbracket z_1 \rrbracket_\gamma \widehat{\gamma} \dagger \widehat{x}(\llbracket z_2 \rrbracket_\beta\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle)] \\
 & \triangleq & z<\widehat{\frac{z_1}{z_2}}\langle \langle z_1.\gamma\rangle \widehat{\gamma} \dagger \widehat{x}(\langle z_2.\beta\rangle\,\widehat{\beta}\,[x]\,\widehat{y}\,\langle y.\alpha\rangle)] \\
 & \rightarrow & z<\widehat{\frac{z_1}{z_2}}\langle \langle z_2.\beta\rangle\,\widehat{\beta}\,[z_1]\,\widehat{y}\,\langle y.\alpha\rangle]
\end{array}
$$

$$\begin{array}{rcl}
\llbracket(\lambda t.u)v\rrbracket_\alpha & \triangleq & \llbracket\lambda t.u\rrbracket_\gamma\widehat{\gamma} \dagger \widehat{x}(\llbracket v\rrbracket_\beta\,\widehat{\beta}\ [x]\ \widehat{y}\,\langle y.\alpha\rangle) \\
& \triangleq & (\widehat{t}\,(t \odot \langle u.\delta\rangle)\,\widehat{\delta}\cdot\gamma)\widehat{\gamma} \dagger \widehat{x}(\langle v.\beta\rangle\,\widehat{\beta}\ [x]\ \widehat{y}\,\langle y.\alpha\rangle) \\
& \to & (\langle v.\beta\rangle\widehat{\beta} \dagger \widehat{t}(t \odot \langle u.\delta\rangle))\widehat{\delta} \dagger \widehat{y}\langle y.\alpha\rangle \\
& \to & v \odot \langle u.\alpha\rangle
\end{array}$$

### 6.3.2 Encoding the λx-calculus

It is easy to extend the encoding to the calculus of explicit substitution, namely for λx-calculus [BR95, LLD$^+$04]. To do that it suffices to add to the definition of encoding λ-calculus the following:

$$\llbracket M\langle x = N\rangle\rrbracket_\alpha := \mathcal{I}_{<\widehat{\mathcal{I}_2}}^{\widehat{\mathcal{I}_1}}\langle\llbracket N_1\rrbracket_\beta\widehat{\beta} \nwarrow \widehat{x}(x \circledcirc \llbracket M_2\rrbracket_\alpha)]$$

where:

$\mathcal{I} = fv(M) \cap fv(N)$
$\llbracket N_1\rrbracket_\beta = ind(\llbracket N\rrbracket_\beta, \mathcal{I}, 1)$
$\llbracket M_2\rrbracket_\alpha = ind(\llbracket M\rrbracket_\alpha, \mathcal{I}, 2)$

Figure 6.6: Encoding the λx-calculus

These results rely on the interpretation of λx in $\mathcal{X}$ presented in [vBLL05].

In what follows we show that the encoding preserves the reduction of the λx-calculus. We first recall the reduction rules of the λx-calculus:

$$\begin{array}{llcl}
\text{(B)} & (\lambda x.M)N & \to & M\langle x = N\rangle \\
\text{(var)} & x\langle x = N\rangle & \to & N \\
\text{(gc)} & M\langle x = N\rangle & \to & M,\ \ x \notin fv(M) \\
\text{(abs)} & (\lambda y.M)\langle x = N\rangle & \to & \lambda y.(M\langle x = N\rangle),\ \ x \in fv(M) \\
\text{(app)} & (M_1M_2)\langle x = N\rangle & \to & M_1\langle x = N\rangle M_2\langle x = N\rangle
\end{array}$$

Notice that we use the variant $\lambda x_{gc}$ with the rule for garbage collection which has a global criterion.

**Lemma 49 (Simulation of the λx-reduction)**

$$\textit{If}\ \ P \xrightarrow{\lambda x} P' \quad \textit{then} \quad \llbracket P\rrbracket_\alpha \xrightarrow{^*\mathcal{X}}{}^+ (fv(P) \setminus fv(P')) \odot \llbracket P'\rrbracket_\alpha$$

**Proof:** By case analysis and induction on the structure of terms.

- Rule (B) : $(\lambda x.M)N \;\to\; M\langle x = N\rangle$.

$$
\begin{array}{rl}
 & [\![(\lambda x.M)N]\!]_\alpha \triangleq \\
\triangleq & \mathcal{I} \lhd \Big( [\![\lambda x.M]\!]_\gamma \widehat{\gamma} \dagger \widehat{y}([\![N]\!]_\beta \widehat{\beta}\ [y]\ \widehat{z}\,\langle z.\alpha\rangle) \Big) \\
\triangleq & \mathcal{I} \lhd \Big( (\widehat{x}\,(x \circledcirc [\![M]\!]_\delta)\,\widehat{\delta} \cdot \gamma)\widehat{\gamma} \dagger \widehat{y}([\![N]\!]_\beta \widehat{\beta}\ [y]\ \widehat{z}\,\langle z.\alpha\rangle) \Big) \\
\to & \mathcal{I} \lhd \Big( ([\![N]\!]_\beta \widehat{\beta} \dagger \widehat{x}(x \circledcirc [\![M]\!]_\delta))\widehat{\delta} \dagger \widehat{z}\langle z.\alpha\rangle \Big) \\
\to & \mathcal{I} \lhd \Big( [\![N]\!]_\beta \widehat{\beta} \dagger \widehat{x}(x \circledcirc [\![M]\!]_\alpha) \Big) \\
\to & \mathcal{I} \lhd \Big( [\![N]\!]_\beta \widehat{\beta} \nwarrow \widehat{x}(x \circledcirc [\![M]\!]_\alpha) \Big) \\
\triangleq & [\![M\langle x = N\rangle]\!]_\alpha
\end{array}
$$

- Rule (var) : $x\langle x = N\rangle \;\to\; N$.

$$
\begin{array}{rl}
 & [\![x\langle x = \alpha\rangle]\!]_\alpha \triangleq \\
\triangleq & [\![N]\!]_\beta \widehat{\beta} \nwarrow \widehat{x}[\![x]\!]_\alpha \\
\triangleq & [\![N]\!]_\beta \widehat{\beta} \nwarrow \widehat{x}\langle x.\alpha\rangle \\
\to & [\![N]\!]_\beta \widehat{\beta} \dagger \widehat{x}\langle x.\alpha\rangle \\
\to & [\![N]\!]_\alpha
\end{array}
$$

- Rule (gc) : $M\langle x = N\rangle \;\to\; M,\ \ x \notin fv(M)$.

$$
\begin{array}{rl}
 & [\![M\langle x = N\rangle]\!]_\alpha \triangleq \\
\triangleq & \mathcal{I}_{<\widehat{\mathcal{I}_2}}^{\widehat{\mathcal{I}_1}}\langle [\![N]\!]_\beta \widehat{\beta} \nwarrow \widehat{x}(x \odot [\![M]\!]_\alpha)],\ \text{where } \mathcal{I} = fv(N) \cap fv(M) \\
\to & \mathcal{I}_{<\widehat{\mathcal{I}_2}}^{\widehat{\mathcal{I}_1}}\langle (fv(N)) \odot [\![M]\!]_\alpha] \\
\dashrightarrow & (fv(N) \setminus fv(M)) \odot [\![M]\!]_\alpha
\end{array}
$$

- Rule (abs) : $(\lambda y.M)\langle x = N\rangle \;\to\; \lambda y.(M\langle x = N\rangle),\ \ x \in fv(M)$.

$$
\begin{array}{rl}
 & [\![(\lambda y.M)\langle x = N\rangle]\!]_\alpha \triangleq \\
\triangleq & \mathcal{I}_{<\widehat{\mathcal{I}_2}}^{\widehat{\mathcal{I}_1}}\langle [\![N]\!]_\beta \widehat{\beta} \nwarrow \widehat{x}[\![\lambda y.M]\!]_\alpha],\ \text{where } \mathcal{I} = fv(N) \cap fv(M) \\
\to & \mathcal{I}_{<\widehat{\mathcal{I}_2}}^{\widehat{\mathcal{I}_1}}\langle (fv(N)) \odot [\![M]\!]_\alpha] \\
\dashrightarrow & (fv(N) \setminus fv(M)) \odot [\![M]\!]_\alpha
\end{array}
$$

- As for the rule (app), we would have to modify it in a way that it checks if $x$ appears in both $M_1$ and $M_2$ before unnecessarily duplicating the explicit substitution. The proof is then straightforward. □

### 6.3.3 Encoding the λlxr-calculus

Furthermore, it is possible to encode λlxr-calculus [KL07], a calculus which can be seen as an extension of λx with explicit terms corresponding to weakening and contraction (see Section 4.2, page 28). Due to the linearity constraints in the λlxr-calculus its interpretation in λlxr is somewhat simpler:

$$\begin{aligned}
\llbracket x \rrbracket_\alpha &:= \langle x.\alpha \rangle \\
\llbracket \lambda x.t \rrbracket_\alpha &:= \widehat{x}\, \llbracket t \rrbracket_\beta\, \widehat{\beta} \cdot \alpha \\
\llbracket tu \rrbracket_\alpha &:= \llbracket u \rrbracket_\gamma \widehat{\gamma} \dagger \widehat{y}(\llbracket t \rrbracket_\beta\, \widehat{\beta}\, [y]\, \widehat{z}\, \langle z.\alpha \rangle) \\
\llbracket t\langle x = u \rangle \rrbracket_\alpha &:= \llbracket u \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{x} \llbracket t \rrbracket_\alpha \\
\llbracket \mathrm{W}^x(t) \rrbracket_\alpha &:= x \odot \llbracket t \rrbracket_\alpha \\
\llbracket \mathrm{C}_z^{x,y}(t) \rrbracket_\alpha &:= z{<}^{\widehat{x}}_{\widehat{y}}\langle \llbracket t \rrbracket_\alpha \rangle
\end{aligned}$$

Figure 6.7: Encoding the λlxr-calculus

In particular, the terms representing weakening and contraction are encoded in a natural way, which is not a surprise, since λlxr-calculus was one of the sores of inspiration for ${}^*\mathcal{X}$ calculus.

#### Simulation of λlxr-reduction

If we assume the freedom to permute the independent erasers and duplicators with other terms in ${}^*\mathcal{X}$, we can encode easily most of the reduction rules of λlxr-calculus. We can not encode the rule Comp (see Section 4.2.3, page 31) which represents the composition of explicit substitutions, and in our understanding would correspond to allowing active cuts to pass over other active cuts, which we do not have in ${}^*\mathcal{X}$.

Nevertheless it is important to show the relation between the reducing in these two calculi. We denote by $\xrightarrow{\lambda\mathsf{lxr}'}$ the reduction relation which is obtained by excluding the rule Comp from the set of reduction rules of the λlxr-calculus.

**Lemma 50 (Simulation of the λlxr-reduction)**

$$\textit{If}\ \ P \xrightarrow{\lambda\mathsf{lxr}'} P' \quad \textit{then} \quad \llbracket P \rrbracket_\alpha \xrightarrow{{}^*\mathcal{X}} {}^* \llbracket P' \rrbracket_\alpha$$

**Proof:** By case analysis and induction on the structure of terms. We give the proof for the rules: $\mathsf{Weak}_1$, $\mathsf{Cont}$ and $\mathsf{Merge}$ which appear in both calculi.

- Rule $\mathsf{Weak}_1$ : $\mathrm{W}_x(t)\langle x = u\rangle \longrightarrow \mathrm{W}_{\mathcal{FV}(u)}(t)$.

$$\begin{aligned}
& \llbracket \mathrm{W}_x(t)\langle x = u\rangle \rrbracket_\alpha \triangleq \\
\triangleq\ & \llbracket u \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{x} \llbracket \mathrm{W}_x(t) \rrbracket_\alpha \\
\triangleq\ & \llbracket u \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{x}(x \odot \llbracket t \rrbracket_\alpha) \\
\rightarrow\ & fv(u) \odot \llbracket t \rrbracket_\alpha \\
\triangleq\ & \llbracket \mathrm{W}_{fv(u)}(t) \rrbracket_\alpha
\end{aligned}$$

- Rule $\mathsf{Cont}$ : $\mathrm{C}_x^{y,z}(t)\langle x = u\rangle \longrightarrow \mathrm{C}_{fv(u)}^{\Psi,\Upsilon}(t\langle y = u_1\rangle\langle z = u_2\rangle)$, where $\Psi, \Upsilon$ are fresh lists of names, $u_1 = \mathcal{R}_\Psi^{fv(u)}$, $u_2 = \mathcal{R}_\Upsilon^{fv(u)}$. We assume for simplicity that the renaming operation is performed by indexing, i.e. that $\Psi, \Upsilon$ are in fact $fv(u_1), fv(u_2)$, respectively; where $u_i := ind(u, fv(u), i)$, $i = 1, 2$. Thus we have:

$$\begin{aligned}
& \llbracket \mathrm{C}_x^{y,z}(t)\langle x = u\rangle \rrbracket_\alpha \triangleq \\
\triangleq\ & \llbracket u \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{x} \llbracket \mathrm{C}_x^{y,z}(t) \rrbracket_\alpha \\
\triangleq\ & \llbracket u \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{x}(x{<}\tfrac{\widehat{y}}{\widehat{z}}\langle \llbracket t \rrbracket_\alpha]) \\
\rightarrow\ & fv(u){<}\tfrac{\widehat{fv(u_1)}}{\widehat{fv(u_2)}}\langle \llbracket u_2 \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{z}(\llbracket u_1 \rrbracket_\gamma \widehat{\gamma} \nwarrow \widehat{y} \llbracket t \rrbracket_\alpha)] \\
\triangleq\ & fv(u){<}\tfrac{\widehat{fv(u_1)}}{\widehat{fv(u_2)}}\langle \llbracket u_2 \rrbracket_\beta \widehat{\beta} \nwarrow \widehat{z} \llbracket t\langle y = u_1\rangle \rrbracket_\alpha] \\
\triangleq\ & fv(u){<}\tfrac{\widehat{fv(u_1)}}{\widehat{fv(u_2)}}\langle \llbracket t\langle y = u_1\rangle\langle z = u_2\rangle \rrbracket_\alpha] \\
\triangleq\ & \llbracket \mathrm{C}_{fv(u)}^{\Psi,\Upsilon}(t\langle y = u_1\rangle\langle z = u_2\rangle) \rrbracket_\alpha
\end{aligned}$$

- Rule $\mathsf{Merge}$ : $\mathrm{C}_w^{y,z}(\mathrm{W}_y(t)) \longrightarrow \mathcal{R}_w^z(t)$.

$$\begin{aligned}
& \llbracket \mathrm{C}_w^{y,z}(\mathrm{W}_y(t)) \rrbracket_\alpha \triangleq \\
\triangleq\ & w{<}\tfrac{\widehat{y}}{\widehat{z}}\langle \llbracket \mathrm{W}_y(t) \rrbracket_\alpha] \\
\triangleq\ & w{<}\tfrac{\widehat{y}}{\widehat{z}}\langle y \odot \llbracket t \rrbracket_\alpha] \\
\dashrightarrow\ & \llbracket t \rrbracket_\alpha\{w/z\} \\
\triangleq\ & \llbracket \mathcal{R}_w^z(t) \rrbracket_\alpha
\end{aligned}$$

Thus we are done with the proof. □

# Part II

# Diagrammatic Classical Computing

# Chapter 7

# Introduction

Diagrams and pictures are among the oldest means of human communication. They are usually used for simple representation, but they also have the potential to express more complex features, such as to carry out certain types of reasoning. Therefore they can have an important role in logic and mathematics. However, diagrams are playing a secondary role, since the sentential style of representation was dominant in the history of modern logic. Inference was usually seen as a manipulation with pure symbols according to formal rules.

The ideas about other possible ways to model reasoning existed before, but recently they are becoming more popular, especially the diagrammatic systems. This trend exists not only in logic, but also among philosophers, cognitive scientists and computer scientists. Although its nature is not clear, there should be a relation between the external diagrammatic representations, which interest logicians and computer scientists, and visual images as internal mental representations, which are considered by cognitive scientists and philosophers. Not all scientists agree on the relevance of mental images in the process of thinking, some consider the symbols to play the key role. Among the scientists who were in favor of images over symbols were Albert Einstein[1] and Nikola Tesla[2].

It is important to say that even before the era of modern logic, several well known diagrammatic systems were introduced to illustrate reasoning. For example, Euler diagrams (dating back to 1768, [Eul68]), a system of

[1] "The words or the language, as they are written or spoken, do not seem to play any role in my mechanism of thought.", in [Ghi87].

[2] "When I get a new idea, I start at once building it up in my imagination ... and when I see no fault anywhere, I put into concrete form the final product of my brain.", in [Joh82].

closed curves used to illustrate the syllogistic reasoning, which is at the core of deductive reasoning. We might also mention Venn diagrams, as a related system. Besides that, a very important diagrammatic system for mathematical logic was developed by Frege, in Begriffsschrift (in 1879) [vH67], where he introduces his concept notation.

## Recent developments and related work

*Proof nets*, as a graphical representation of proofs in linear logic, were introduced by Girard [Gir87]. They came as a solution to a problem of describing the identity of proofs in linear logic. Thus, proof nets can be seen as presentation of proofs that is free from the so-called syntactic bureaucracy. They also have a dynamic aspect, the cut-elimination. Although primarily designed for linear logic, their structure is universal and they were adapted for other logics.

The notion of a proof-net is extended to *standard* classical logic by Robinson [Rob03], where some of the issues dating back to [Gir91] were resolved. For example, extending the correctness criterion was done in rather natural way. Classical proof nets were later studied in [LS05], in relation with calculus of structures (see for example [Gug04]). The calculus of structures has also been an inspiration for the three-dimensional representation of proofs, recently presented by Giraud [Gui06]. There has been some earlier work on two-dimensional objects in the classical framework, for example logical flow graphs [Bus91].

In this second part of this thesis, we present a diagrammatic calculus which is closely related to classical proof nets, as formulated in [Rob03]. Robinson arrived to his diagrammatic formulation directly from classical proofs presented in two-sided Gentzen's sequent calculus. On the other hand, we arrived to our formulation of the diagrammatic calculus in three steps. Starting point is the study of classical sequent calculus with explicit structural rules with incorporated notion of linearity, followed by the design of the ${}^*\mathcal{X}$ calculus which captures the structure and the dynamics of such proofs, and then we derive the diagrammatic calculus as a two-dimensional view on ${}^*\mathcal{X}$-terms.

There are some differences between our formulation and that of Robinson. We addopt the convention which says that, if we see diagrams as circuits, entering wires are coming from the left-hand side of the circuit, while the exiting wires are are going towards the right-hand side. That way we have a natural correspondence between free entering/exiting wires

of a circuit and the premise/conclusion formulas in a sequent that is being proved.

The result presented by Robinson can be applied to this work. He provides an analysis of the connection between conventional sequent calculus and diagrams, i.e., proof nets. Moreover, he defines the correctness condition for those nets. His work deals mainly with static aspects, whereas we are more interested in computational aspects of diagrams.

*Interaction nets* are a programming paradigm which represents the generalization of proof nets. They were discovered after an attempt to implement the reduction of proof nets. Interaction nets were introduced and studied by Lafont [Laf90, Laf95b]. In general, a net consists of finite number of *cells* which are labeled, and *ports* through which the cells interact. One port is defined as principal, whilst the others are auxiliary ports. It is assumed that all cells labelled by a same symbol have the same number of ports. The dynamic behavior is specified by *rewriting rules* which are *local*, and which preserve the set of free ports. Moreover, the computation with interaction nets is parallel and asynchronous. Interaction nets are Turing complete.

*Interaction combinators* [Laf97] are a fixed system of interaction nets which consists of three agents (combinators) and only six reduction rules. It has been shown that this rewriting system is universal, that is, it can simulate any other system of interaction nets.

# Chapter 8

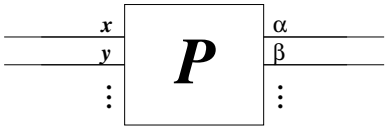
# Diagrammatic calculus: ${}^{d}\mathcal{X}$

This chapter presents the diagrammatic calculus ${}^{d}\mathcal{X}$, with its syntax and reduction rules (including the simplification rules). We formulate the basic operational properties and define the type system for diagrams.

## Introduction

After designing and studying the ${}^{*}\mathcal{X}$ calculus, it was natural to think about the diagrammatic representation of its terms. This is due to a fact that ${}^{*}\mathcal{X}$ is a linear model of computation which introduces explicit terms for implementing erasure and duplication. Therefore we derive from it a two-dimensional calculus for classical logic called ${}^{d}\mathcal{X}$. Although ${}^{*}\mathcal{X}$-terms are not in one-to-one correspondence with ${}^{d}\mathcal{X}$-diagrams, the syntax and reduction rules of the two calculi are closely related.

The basic notion in the framework of ${}^{d}\mathcal{X}$ is *port* and it corresponds to the notion of a *name* in the ${}^{*}\mathcal{X}$ calculus. Ports represent the interface of a diagram. Similarly as for names, there are two kinds of ports - entering ports, called *in-ports*, and exiting, called *out-ports*. Port are labelled by placing a name above the line.[1] We imagine a diagram as a circuit with entering and exiting wires:




[1] Labels on the ports is not essential, they are primarily used to establish intuitive relation with the one-dimensional model.

where $P$ is an actual diagram. By convention in-ports are always coming from the left-hand side of a component, and out-ports are always going towards right-hand side. Latin letters $x, y, z, \ldots$ are used to denote in-ports and Greek $\alpha, \beta, \gamma, \ldots$ to denote out-ports. See the syntax given in Figure 8.1 for the description of constructors.

## 8.1 The syntax

The syntax of the diagrammatic calculus ${}^{d}\mathcal{X}$ is defined inductively in Figure 8.1. It consists of eight basic diagram constructors. Notice that there is only one ground element (the first one) which can not be decomposed. The other syntactic elements are built from smaller diagrams.[2] The system we present here can be naturally extended as to encompass the other classical connectives (see Appendix 8.8).

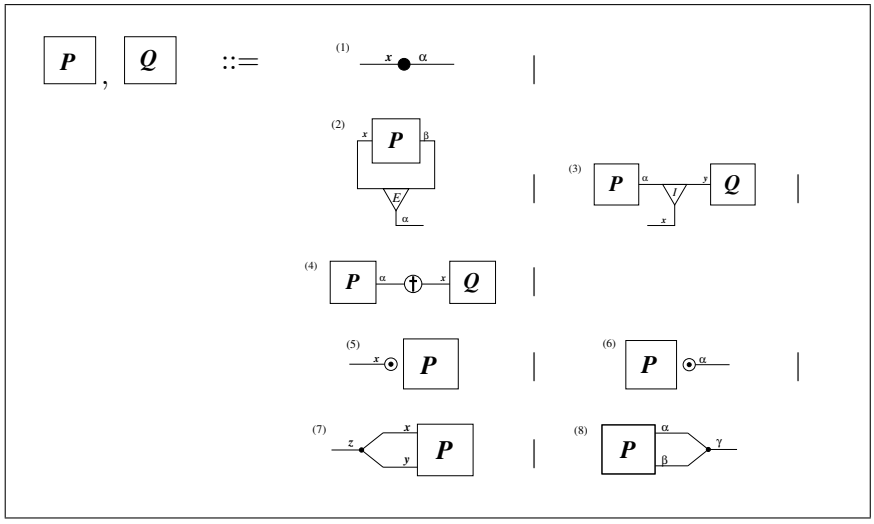
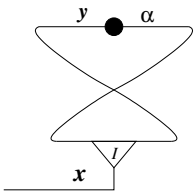



Figure 8.1: The diagrammatic syntax

**Remark** The higher order constructs are always built from smaller components. We assume that, when constructing the diagrams (3) and (4), we use two *not connected* (separate) diagrams $P$ and $Q$. Similarly, in case of diagrams $(2), (5), (6), (7)$ and (8), we assume that $P$ is a unique diagram.

[2]Remark that $\boxed{P}$ should be simple read as - diagram $P$, and does not relate to proof net boxes in linear logic.

**Example 51** *We can not construct, for example an importer, using only one subdiagram. Therefore, we can not obtain a diagram such as this one:*




The diagrams of ${}^{d}\mathcal{X}$ calculus are directly inspired by ${}^{*}\mathcal{X}$-terms, and can be seen as their two-dimensional view. There is however more to this, and the relation between these two approaches is studied in Part III of this thesis. The terminology we use to denote diagrams is different to the one for ${}^{*}\mathcal{X}$ (see Figure 8.3). This is because they are principally different, as one diagram represents a class of terms.

*Convention on ports* We assume the convention which says that the ports of a diagram in general are explicitly presented only when they are relevant. For example we always present the ports which are involved in the process of reducing. The presence of other ports may be assumed.

We assume that ports which are used for building higher order diagrams from smaller diagrams do exist. In what follows we introduce terminology used for diagrams and comment on the element of the syntax.

(1) The first element listed is called *in-out*. It is a ground element and it corresponds to an axiom rule in the sequent calculus. This diagram introduces one free in-port (here $x$) and one free out-port (here $\alpha$).

(2) An *$\mathcal{E}$-fan* is built upon a diagram $P$, by taking an in-port $x$ and an out-port $\beta$ and creating another free out-port $\alpha$.

(3) An *$\mathcal{I}$-fan* is constructed from two not connected diagrams, $P$ and $Q$, linked over their ports $\alpha$ and $y$ respectively, and creating a new in-port $x$.

(4) A *cut* is built from two separate diagrams, $P$ and $Q$, over their ports $\alpha$ and $x$ respectively, and the link obtained in such way contains a dagger (†). The process of computation proceeds towards elimination of daggers from diagrams. It is the only diagram which does not introduce new ports.

(5) A *left-eraser* corresponds to a left-weakening inference rule in the sequent calculus. It enables us to add a fresh in-port (here $x$) to an arbitrary term $P$.

(6) A *right-eraser* is symmetric to left-eraser. It originates from a right weakening rule in the sequent calculus. It enables us to add a fresh out-port (here $\alpha$) to an arbitrary diagram $P$.

(7) A *left-duplicator* corresponds to a left contraction. It takes two in-ports of the diagram $P$ (here $x, y$) and merges them into one (here $z$).

(8) A *right-duplicator* is related to a right contraction rule in the sequent calculus. It represents a merging of two out-ports (here $\alpha, \beta$), into one (here $\gamma$).

**Definition 52 (Ports)** *The set of ports $\mathcal{P}$ of a diagram is defined inductively as presented in Figure 8.2.*

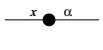
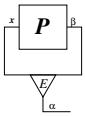
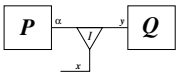
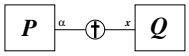
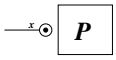
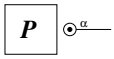
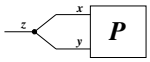
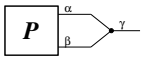
| Diagram | Ports ($\mathcal{P}$) |
|---|---|
|  | $\{x, \alpha\}$ |
|  | $\{\alpha\} \cup \mathcal{P}(P)$ |
|  | $\{x\} \cup \mathcal{P}(P) \cup \mathcal{P}(Q)$ |
|  | $\mathcal{P}(P) \cup \mathcal{P}(Q)$ |
|  | $\{x\} \cup \mathcal{P}(P)$ |
|  | $\{\alpha\} \cup \mathcal{P}(P)$ |
|  | $\{z\} \cup \mathcal{P}(P)$ |
|  | $\{\gamma\} \cup \mathcal{P}(P)$ |

Figure 8.2: The ports

**Definition 53 (Logical and structural ports)** *A port is said to be* logical *if it is created by an in-out, $\mathcal{I}$-fan, or an $\mathcal{E}$-fan.* Structural ports *are*

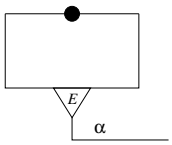
*created by black holes and forks.*[3]

**Lemma 54** *Each diagram has at least one logical out-port.*

**Proof:** By routine induction on the structure of ports. □

The diagrams with minimal number of ports, i.e., with no in-ports and a single out-port are of interest, and we will call them *closed diagrams.*

**Example 55** *The simplest example of such a closed diagram is presented here:*




**Terminology** For clarity we summarize the terminology used.

| ${}^{d}\mathcal{X}$-*diagrams* | ${}^{*}\mathcal{X}$-*terms* | *Logic* |
|---|---|---|
| in-out | capsule | axiom |
| $\mathcal{E}$-fan | exporter | →R-intro |
| $\mathcal{I}$-fan | importer | →L-intro |
| fork | duplicator | contraction |
| black hole | eraser | weakening |
| dagger | cut | cut |
| port | name | named proposition |

Figure 8.3: The terminology

## 8.2 The reduction rules

In this section we formulate the set of reduction rules over diagrams introduced by the syntax.

Reduction rules are divided into four main groups. These are *logical* and *structural* actions and also *activation* and *deactivation* rules. Each group (except logical) can itself be split into two symmetric left and right

[3]This terminology is used because forks and black holes code the so-called *structural* sequent calculus rules, whilst other constructors code *logical* inference rules.

subgroups, in the sense of direction. The way of grouping suggests that symmetries are very common in the calculus. Furthermore, the activation and deactivation groups are dual.

The diagrammatic calculus has a restricted number of reduction rules in comparison with ${}^{*}\mathcal{X}$ calculus. Namely the group called "propagation rules" does not exist in this framework. The basic purpose of this group of reduction rules in ${}^{*}\mathcal{X}$ was to propagate the cut through the structure of terms, until it reaches a point where the propagation ends. In general, propagation ends in the moment when a module, for example $\widehat{\alpha} \nearrow \widehat{x}Q$, reaches an actual place where a free name $\alpha$ occurs in the syntactic representation. In ${}^{d}\mathcal{X}$, since we are in two-dimensional space, there is no need for propagations.

The reduction procedure is directed towards the elimination of †'s (daggers). It captures the essential features of the classical cut-elimination, like for proof nets for linear logic [Gir87]. Therefore it is *non-deterministic* and *non-confluent*, like classical cut-elimination.

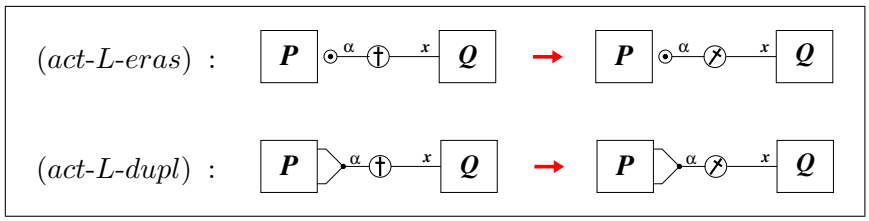
Following the *convention on ports*, we draw only relevant ports in reduction rules, i.e., the ones which are involved in reducing, or created by the reduction rule. The existence of other ports may be assumed.

Following the reduction rules of ${}^{*}\mathcal{X}$ calculus, and the way they are formulated, we formulate the rules of the diagrammatic calculus.

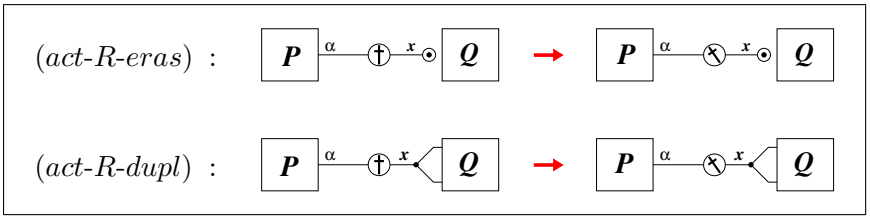
### 8.2.1 Activation rules

This group of rules captures the non-deterministic choice, i.e., the rules define the conditions for the so-called activation of the dagger. This choice is one of the sources of non-confluence in the calculus. The direction of activation is expressed by bending (or activating) the dagger. If the dagger is bent to the position $\nearrow$ (left-dagger), it means that the future actions will be performed towards the left-hand side, and similarly for $\nwarrow$ (right-dagger). There are four possible activations:

- **Left-activation.** *Left-activation* rules define when the dagger can be left-activated. Thus we have:

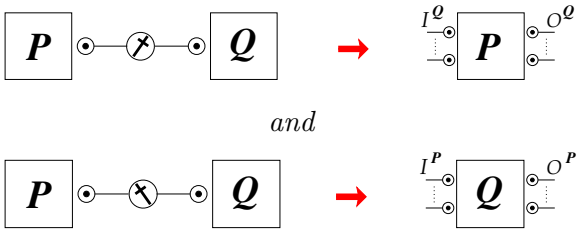


- **Right-activation.** *Right-activation* rules define when the dagger can be right-activated. Thus we have:




Notice that it is not always possible to choose the side of activation. But, having the situation when both directions of activation are possible, is a potential source of non-confluence in the calculus.

**Example 56** *This brings us back to Lafont's example which can be diagrammatically represented as:*

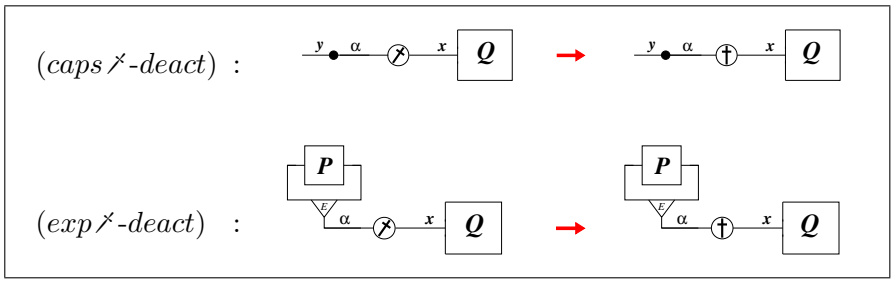



Active cuts are introduced into the system to express non-deterministic choice. We assume that as soon as a dagger is activated, it is treated by some other reduction rule. It was allowed in $\mathcal{X}$ calculus to have more active cuts in a term, but a restriction was introduced, saying that an active cut can not be propagated over another active cut. Otherwise, it would break strong normalization. Since there is no explicit propagation in ${}^{d}\mathcal{X}$, keeping active cuts non-evaluated would not have a particular meaning.

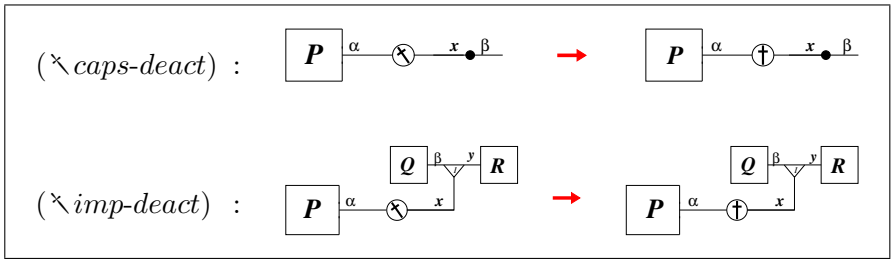
### 8.2.2 Deactivation rules

Deactivation rules define when active dagger can be deactivated. In other words, they are bringing ↗ and ↖ into a vertical position. This group is dual to activation rules.

- **Left-deactivation.** *Left-deactivation* rules define when we are supposed to deactivate a left-dagger.




- **Right-deactivation.** *Right-deactivation* defines when we should deactivate a right-dagger.



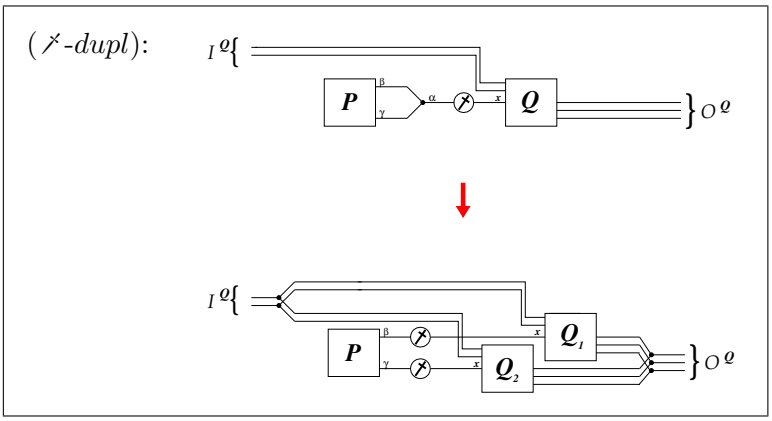

The duality between activation and deactivation rules exists in a sense that by negating the condition and changing the direction of an arrow in the activation rules, we get deactivation rules, and vice versa. A more general formulation of these rules makes the duality more obvious. This is shown in Section 8.3.

### 8.2.3 Structural actions

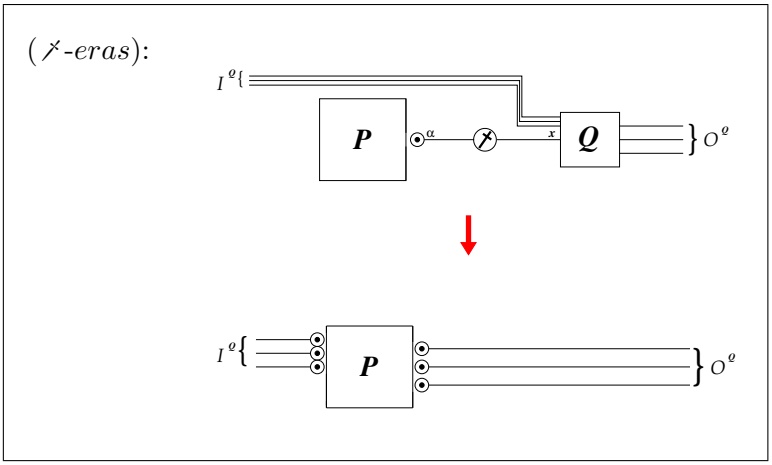
Structural actions define *duplication* and *erasure* of diagrams. These operations are implemented using new constructors, *forks* and a *black holes*[4]. We have the pairs of symmetric rules: *(↗-erasure)* and *(↖-erasure)*, and *(↗-duplication)* and *(↖-duplication)*.

- **Left structural actions.** The rule *(↗-duplication)* defines an action which occurs when a left-activated dagger faces a right fork. The result is the duplication of a term on the right-hand side of a dagger. This is performed in a way that preserves a set of both in-ports and out-ports. This preservation of ports is also implemented using forks. The *(↗-duplication)* is presented here:

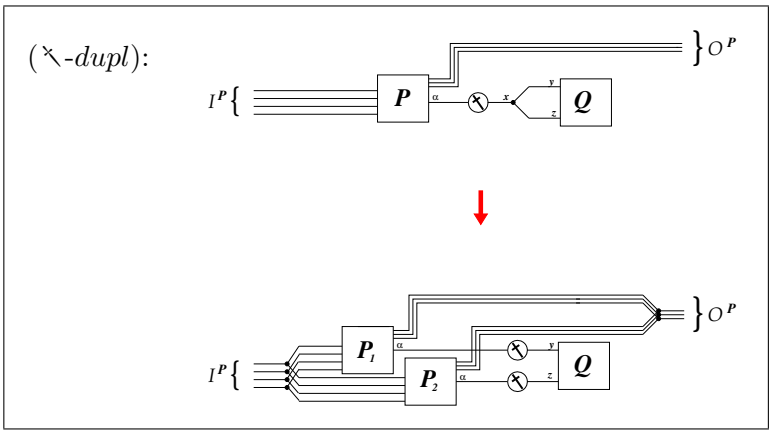
[4]They correspond to an eraser and a duplicator in $^{*}\mathcal{X}$, and to weakening and contraction in sequent calculus.



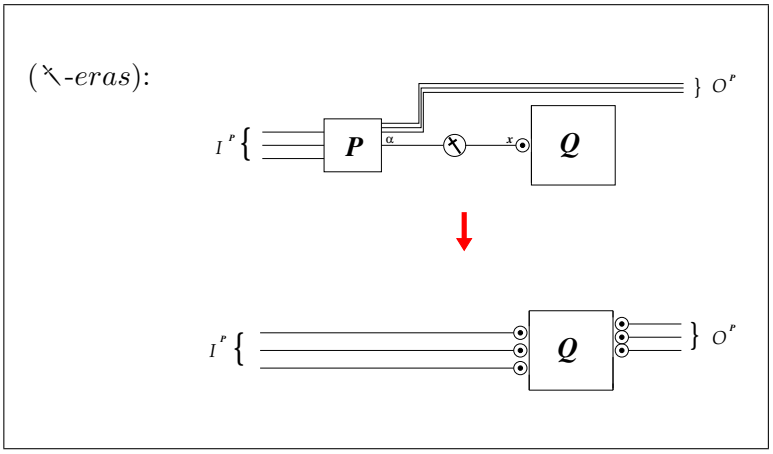

The rule *(↗-erasure)* defines a computational step performed in the situation when left-active cut is facing right black hole. As presented bellow, the term on the right-hand side (here $Q$) is being erased, but the ports are preserved. The ports preservation is implemented using black holes, namely by attaching them to the remaining term (here $P$). The *(↗-erasure)* is presented here:




- **Right structural actions.** The rule *(↖-duplication)* is symmetric with respect to *(↗-duplication)*. It is given by the following diagram modification:




The rule *(↖-erasure)* is symmetric to the *(↗-erasure)* and it is given by the following diagram reduction rule

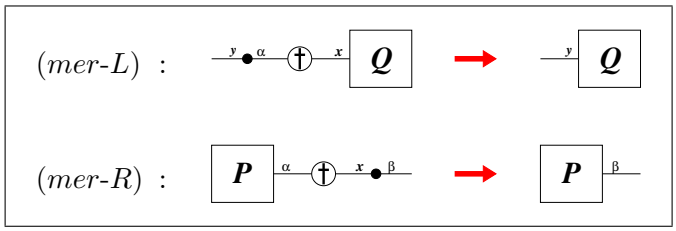



Structural reduction rules are not local transformations as they involve duplication or erasure of arbitrary parts of digrams. This also stands for proofs in the sequent calculus, where parts of proofs are being erased or duplicated in a similar manner. Moreover they are non-deterministic, in the sense that the part of a diagram duplicated or deleted is not uniquely determined.

### 8.2.4 Logical actions

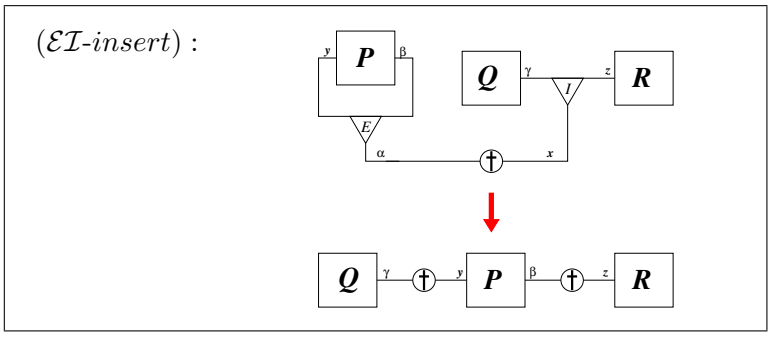
Logical actions define the way to eliminate the daggers when logical ports are involved.

- **Merging.** The first two rules specify merging of an *in-out* with another diagram:




Notice that diagrams $P$ and $Q$ are arbitrary. This means that they are not necessarily connected to an in-out over a logical port.

- **Inserting.** The last rule is called *insertion* and describes how to reduce a diagram that connects an $\mathcal{E}$-fan with an $\mathcal{I}$-fan through a dagger. The term $P$ is inserted between the terms $Q$ and $R$. Notice that one dagger is substituted by two more atomic daggers.




## 8.3 Generalization of activation and deactivation rules

In this section we show that the activation and deactivation rules of ${}^{d}\mathcal{X}$ calculus, can be presented in a more general way, by using the notions of L-principal and S-principal name. That way the duality activation/deactivation becomes obvious. Thus for activation we have:

**Left-activation:**


when $\alpha$ is S-principal for $P$.

**Right-activation:**


when $x$ is S-principal for $Q$.

On the other hand, for deactivation rules we have:

**Left-deactivation:**


when $\alpha$ is L-principal for $P$.

**Right-deactivation:**


when $x$ is L-principal for $Q$.

Notice that the condition "$\alpha$ is not L-principal for $P$" is equivalent to "$\alpha$ is S-principal for $P$". The duality is in a sense that if we negate the condition and reverse an arrow of a rule, we get a rule from a dual group.

## 8.4 Diagram simplification

In this section we define the *simplification rules*, denoted by $\dashrightarrow$. They are used to simplify diagrams when it is possible, but they are not reduction rules as they do not involve cuts. Simplification rules are given in Figure 8.4.

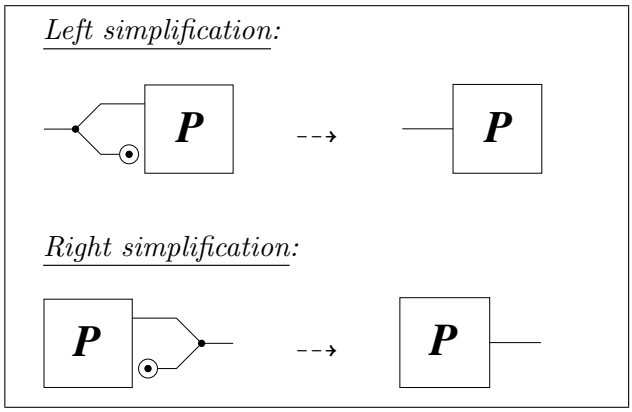
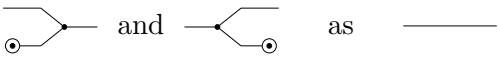
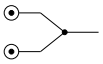
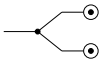



Figure 8.4: Diagram simplification

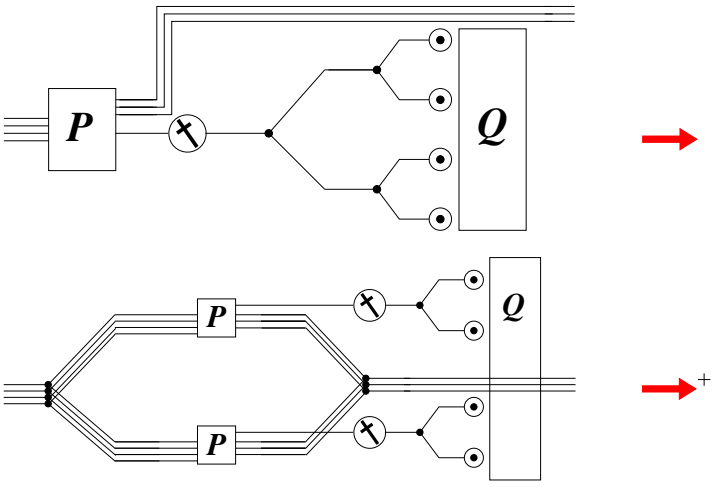
These rules basically state that the construction such as fork applied to a black hole, does not have a computational meaning with respect to reduction procedure. Furthermore, the idea to rewrite in an arbitrary diagram all branches of the form

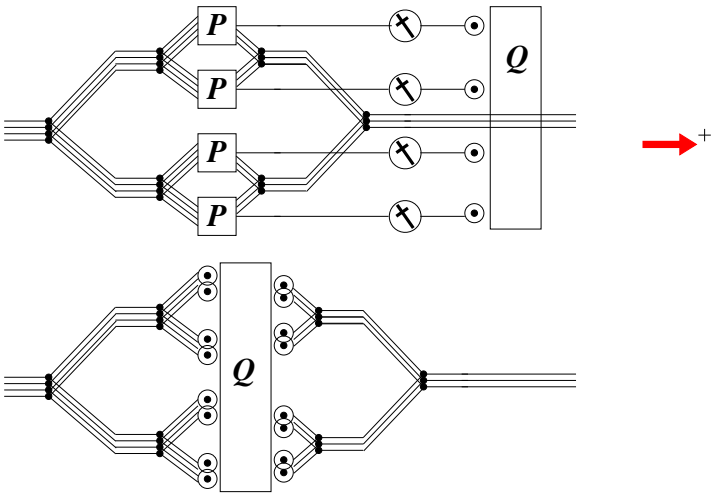



is also motivated by optimizing computation, in a sense that we do not wish to perform numerous duplications of a term, if we are going to erase all those copies afterwards. This situation is avoided by introducing the simplification rules. They are performed with priority, that is, whenever possible before and during the computation.

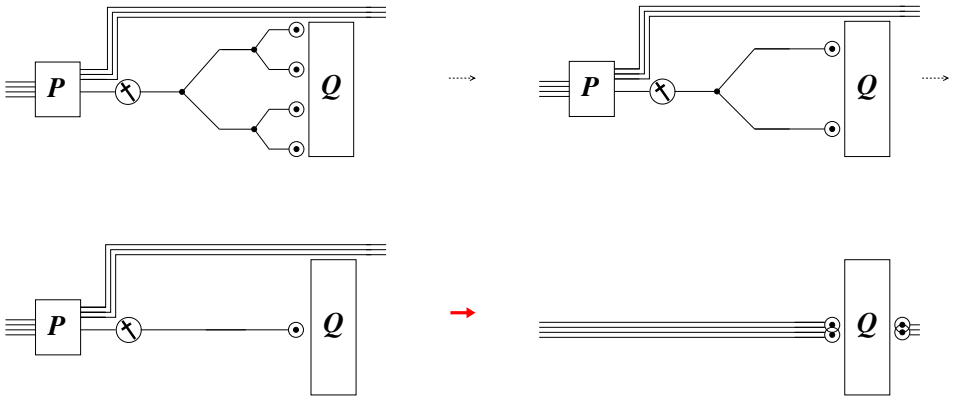
When both branches of a fork end in black holes gives a special case of the previous. For the right fork we have:  is rewritten by  , and similarly for the left fork:  is rewritten by  .

**Example 57** *The following example illustrates the difference between, on the one hand, computation with simplification and, on the other hand, computation without the use of simplification. Observe first the sequence of reductions:*






*The diagram $P$ is duplicated two times, and then all four copies are erased. Notice that in the result, as a sideproduct, we get a certain number of forks applied to black holes. Instead of dealing with this complex diagram modifications, by adopting the simplification rules, our approach is the following:*




Simplification rules can be seen as a kind of *garbage collection*.

## 8.5 Operational properties

The reduction system in ${}^{d}\mathcal{X}$ satisfies the property of interface preservation, that is, preservation of the set of ports.

**Theorem 58 (Preservation of ports)** *The set of ports is preserved by reductions and simplification rules:*

1. *If* $P \to Q$ *then* $\mathcal{P}(P) = \mathcal{P}(Q)$
2. *If* $P \dashrightarrow Q$ *then* $\mathcal{P}(P) = \mathcal{P}(Q)$

**Proof:** This property can be easily checked by inspection of reduction rules and simplification rules. □

## 8.6 The typing rules

In this section we show how to assign types to diagrams. It is done in way similar to that for ${}^*\mathcal{X}$-terms.

Given a set $T$ of basic types, a type is given by

$$A, B ::= T \mid A \to B.$$

The type assignment of a diagram $P$ is given by an expression $P :\cdot\ \Gamma \vdash \Delta$. Here $\Gamma$ stands for a set of type declarations for in-ports, while $\Delta$ stands for a set of declarations for out-ports. If $P$ is a diagram, an expression

$$P :\cdot\ \Gamma \vdash \Delta$$

is used to denote a *type assignment*. The type system is presented in Figure 8.5.

**Theorem 59 (Subject reduction)**

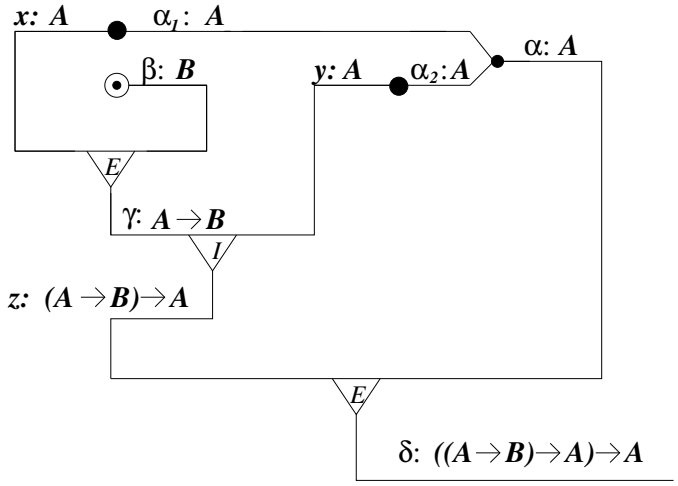
1. *If* $D_1 :\cdot\ \Gamma \vdash \Delta$ *and* $D_1 \to D_2$ *then* $D_2 :\cdot\ \Gamma \vdash \Delta$
2. *If* $D_1 :\cdot\ \Gamma \vdash \Delta$ *and* $D_1 \dashrightarrow D_2$ *then* $D_2 :\cdot\ \Gamma \vdash \Delta$

**Proof:** The proof goes similarly to that in the one-dimensional framework (Theorems 31 and 32 on page 63). □

**Example 60** *As an example we will take the diagram which represents the proof for Peirce's law. Thus we revisit the example on page 60, where this was studied in one-dimensional case. We recall here the term representing the Peirce's law:*

$$\widehat{z}\,([(\widehat{x}\,(\langle x.\alpha_1\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\gamma}\ [z]\ \widehat{y}\,\langle y.\alpha_2\rangle\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}} > \alpha)\,\widehat{\alpha}\cdot\delta$$

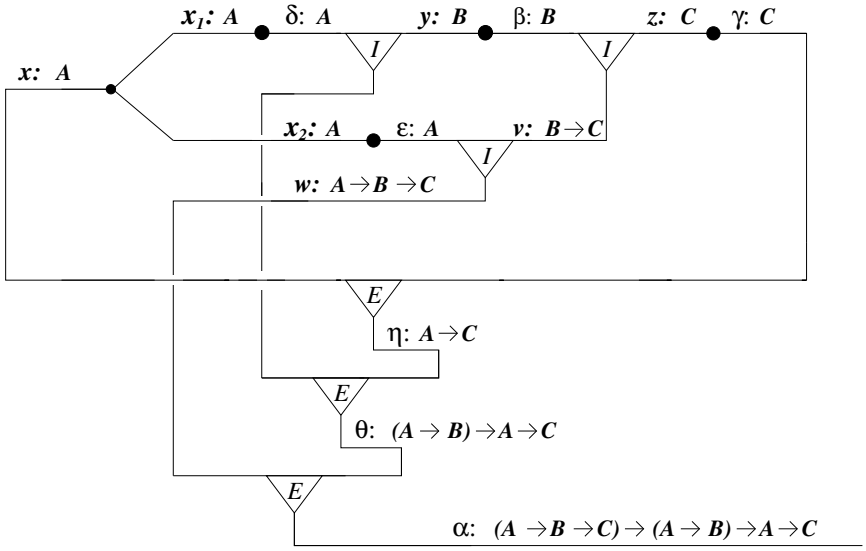
*The diagram with types is represented as follows:*




**Example 61** *Another example is a diagram for the S-combinator. We recall the* ${}^{*}\mathcal{X}$*-term which corresponds to this operator (given in the example on page 61):*

$$\widehat{\omega}\,(\widehat{u}\,(\widehat{x}\,(x{<}\overset{\widehat{x_1}}{\widehat{x_2}}\langle\langle x_2.\epsilon\rangle\,\widehat{\epsilon}\,[w]\;\widehat{v}\,(\underline{\underline{(\langle x_1.\delta\rangle\,\widehat{\delta}\,[u]\;\widehat{y}\,\langle y.\beta\rangle)}}\,\widehat{\beta}\,[v]\;\widehat{z}\,\langle z.\gamma\rangle)])\,\widehat{\gamma}\cdot\eta)\,\widehat{\eta}\cdot\theta)\,\widehat{\theta}\cdot\alpha$$

*The typed diagram representing this term is as follows:*



## 8.7 Implementing data types

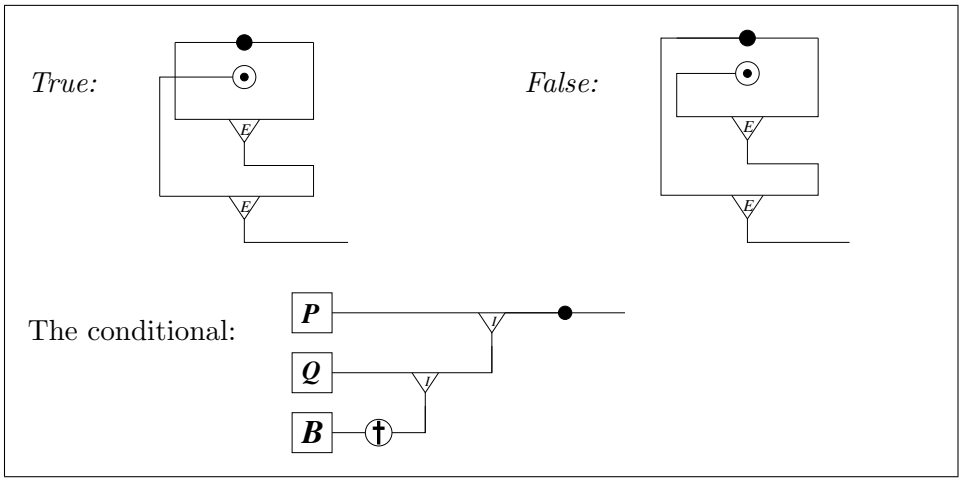
In this section we show how the diagrammatic calculus ${}^{d}\mathcal{X}$ can be used to implement some standard data types. This is needed if we wish to think of the ${}^{d}\mathcal{X}$ calculus as a programming language. The implementation is done in an analogous way as for the ${}^{*}\mathcal{X}$ calculus.

### 8.7.1 Booleans

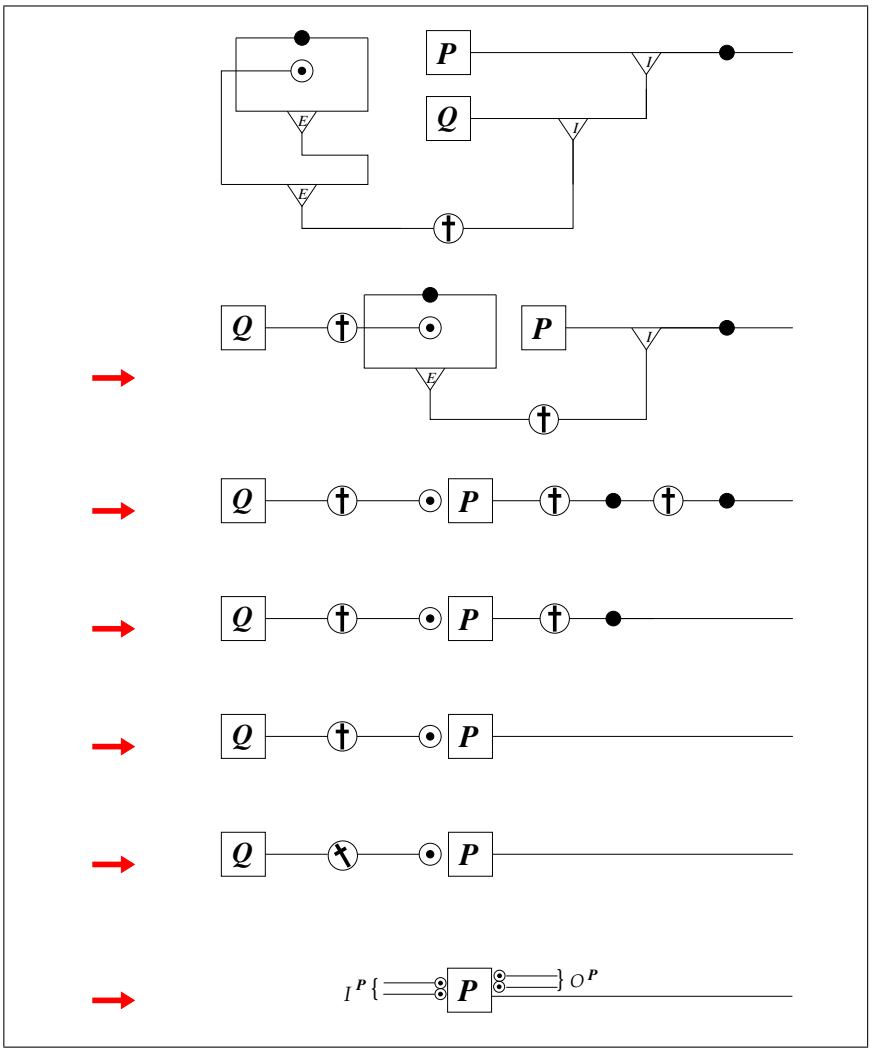
In a programming language a very common construction is a *conditional* (or *if*). We show here how it is implemented in ${}^{d}\mathcal{X}$. A conditional should be a construction with two places to plug in arbitrary diagrams, and a third place to plug in a boolean. The behavior we expect is the following:

- if we plug in two diagrams, $P$ and $Q$, and a boolen $True$, the result of the computation should be $P$, otherwise,
- if we plug in $P$, $Q$ and $False$, the result should be $Q$.

Because of the non-confluence of the calculus, we have to assume that $P$ and $Q$ are closed diagrams, i.e., that they have only one port - a logical out-port. The other solution would be to introduce priority on either left or right-activation. The booleans are defined in the continuation.



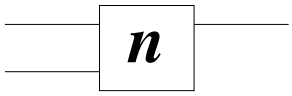

**Example 62** *The computation, when $B = True$ and $P, Q$ are closed diagrams, yields the result $P$. This was already studied in the one-dimensional framework, and thus we are revisiting the example given on page 71. The steps of the computation are given here:*



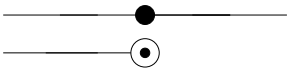

*The order of applying reduction rules, i.e., the order of eliminating cuts is not relevant in this example. The computation goes similarly if we take $B = False$, yielding the diagram $Q$ as a result.*

### 8.7.2 Natural numbers

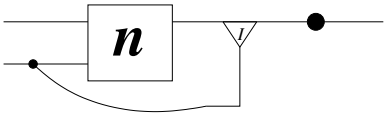
The diagrammatic representation of natural numbers is rather simple and intuitive. Every natural number is a diagram with two in-ports and a single outport. Numbers $n$ and zero are represented as:



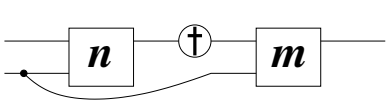

and

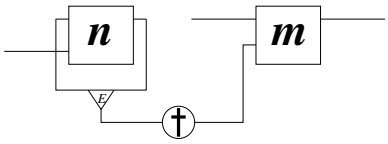


The *successor* ($succ(n)$) of a natural number $n$ is defined as presented by the diagram:




*Addition* ($add(n, m)$) and *multiplication* ($mul(n, m)$) are implemented in an elegant way, as follows. The first diagram represents an addition of natural numbers $m$ and $n$, and the second is multiplication.



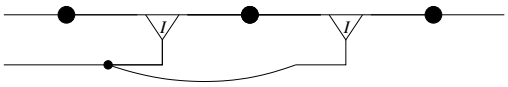




**Example 63** *To provide the reader with more intuition, we illustrate how these operations work. In what follows we list several very simple examples.*

- $succ(0)$*:*

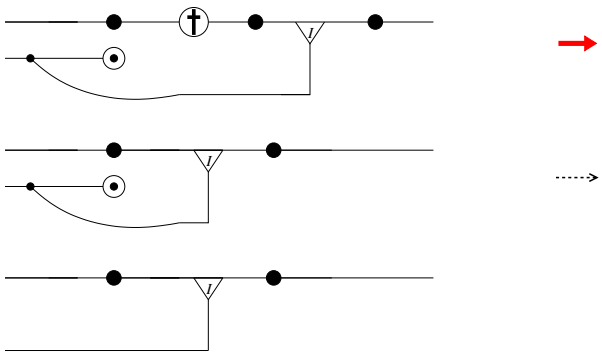

*Notice that the computation is done by applying only a simplification rule; the result is* 1.

- $succ(1)$*:*



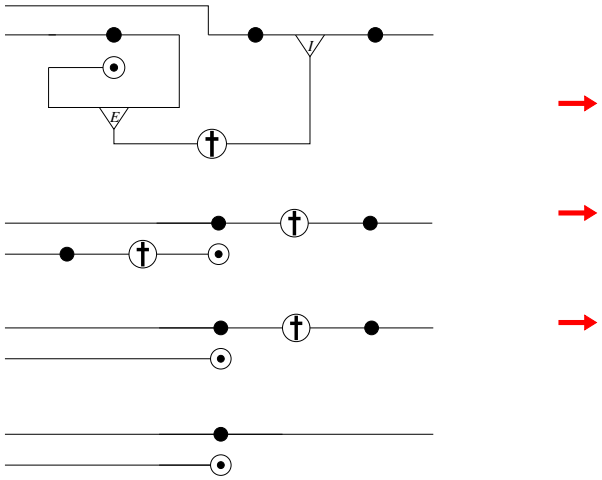

*This is by definition.*

- $add(0, 1)$*:*



*The result is computed by applying one reduction and one simplification rule; the resulting diagram corresponds to* 1.

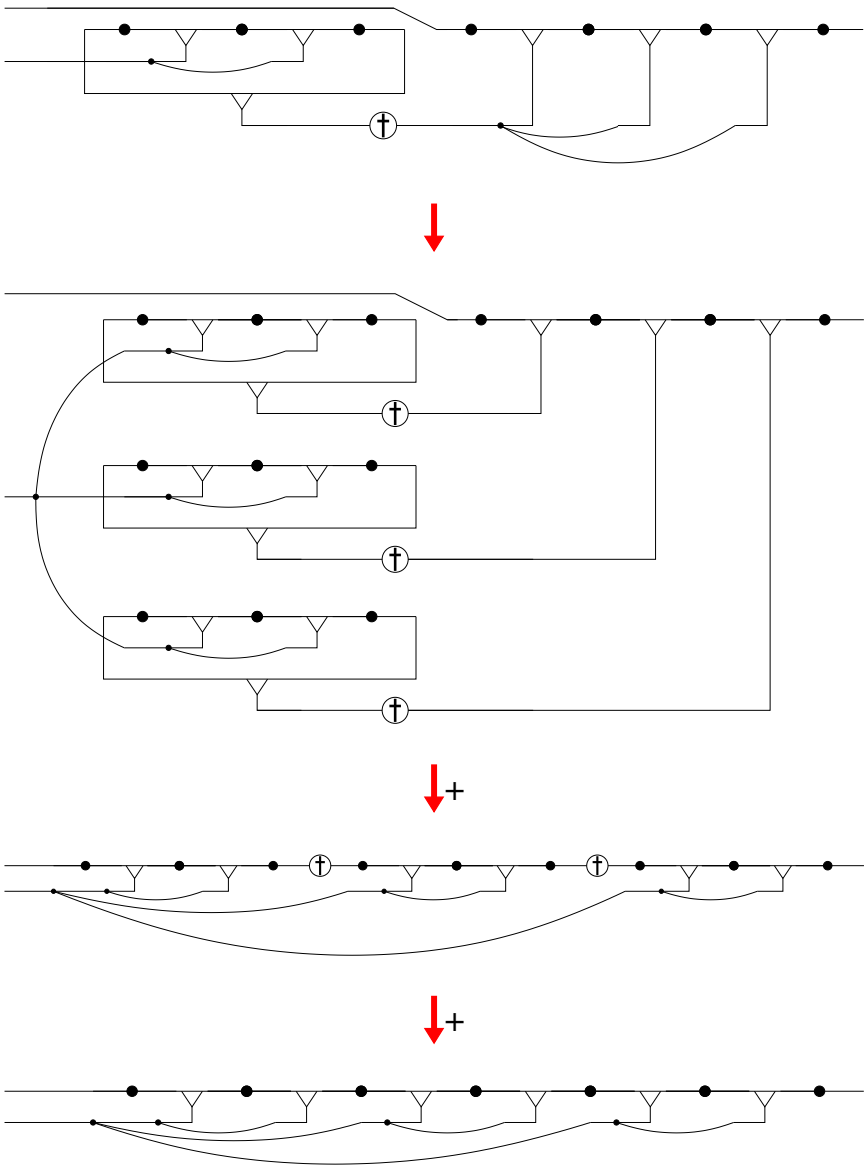
- $times(0,1)$*:*



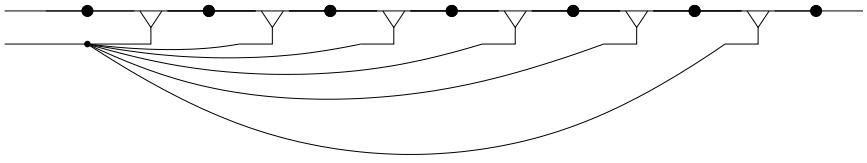
*The result is computed by applying reduction rules; the resulting diagram is zero.*

Thus the natural numbers are composed of $\mathcal{I}$-fans, in-outs and forks. Notice that a natural number $N = n$ consists of $n$ $\mathcal{E}$-fans (and also $n + 1$ in-outs and $n$ left forks).

**An example of computing** We study the computation by taking an example $2 \times 3 = 6$. In some cases several reductions can be (and are) applied simultaneously, which suggests that the computation is essentially parallel. The reader may also notice that the diagram transformations are non-local, because of the duplication which takes place during computation.



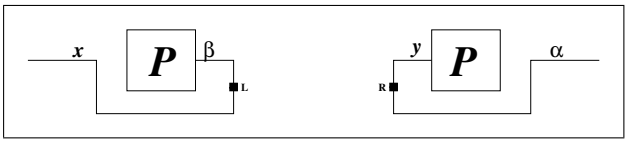

We can simplify the representation of the last diagram. Thus we obtain:



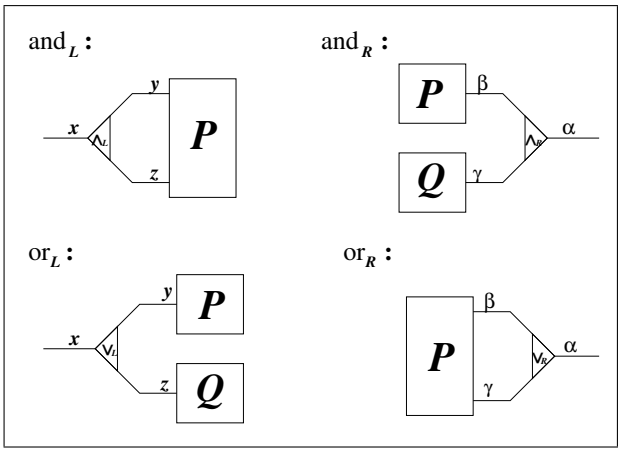
which, as expected, is a diagrammatic representation of natural number 6.

## 8.8 Extension of the ${}^{d}\mathcal{X}$ calculus

**The syntax** First we extend the syntax of ${}^{d}\mathcal{X}$ (given in Figure 8.1 on page 104) with new diagrammatic constructors. Negations are the most important. Left negation is represented by a diagram which simply transforms an out-port into an in-port. Right negation does the opposite. The constructors $\text{not}_L$ and $\text{not}_R$ are given respectively:



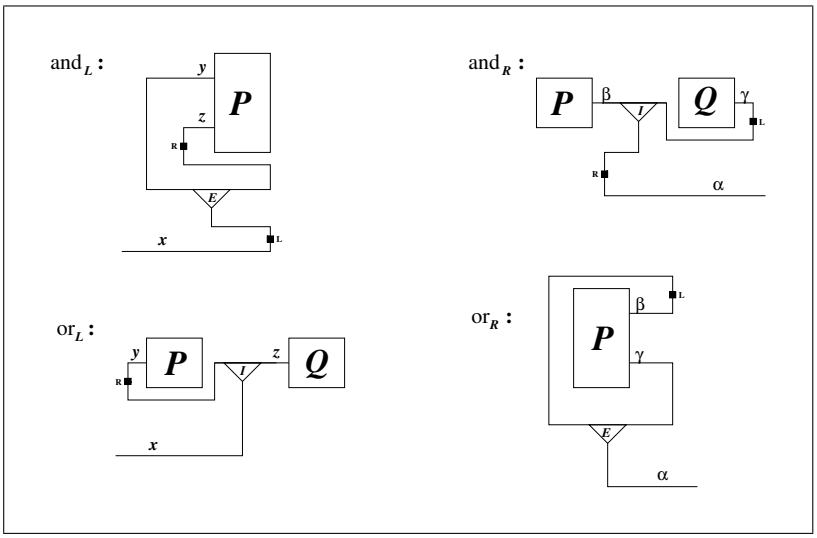

The other constructors: $\text{and}_L$, $\text{and}_R$, $\text{or}_L$ and $\text{or}_R$ are represented as follows:




Adding only the negation to the system is sufficient to express other connectives. We illustrate how conjunctions and disjunctions can be represented by negation and implication; by using the formulas

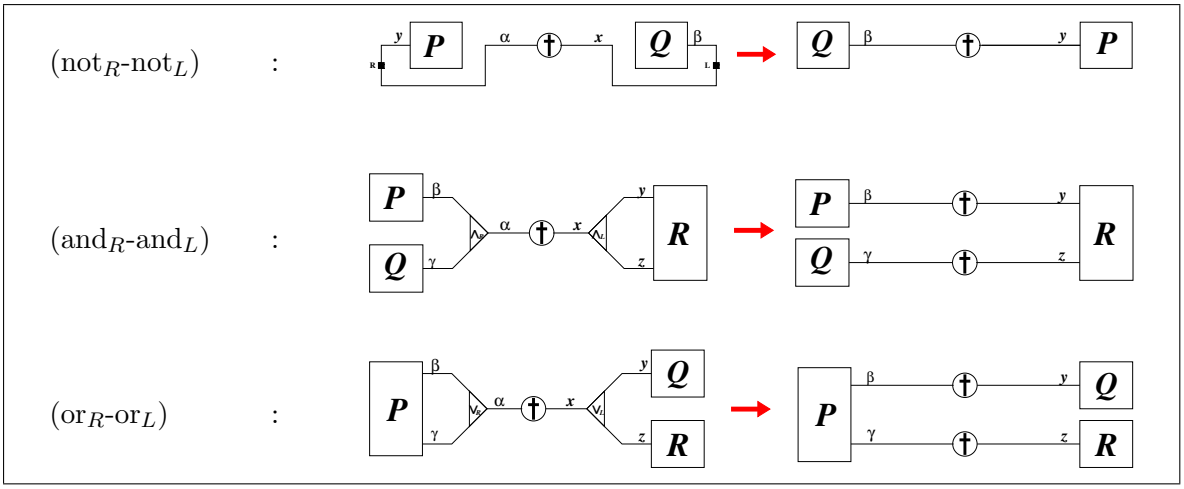
$$A \wedge B \;\leftrightarrow\; \neg(A \rightarrow \neg B) \quad \text{and} \quad A \vee B \;\leftrightarrow\; \neg A \rightarrow B$$

Thus we have:

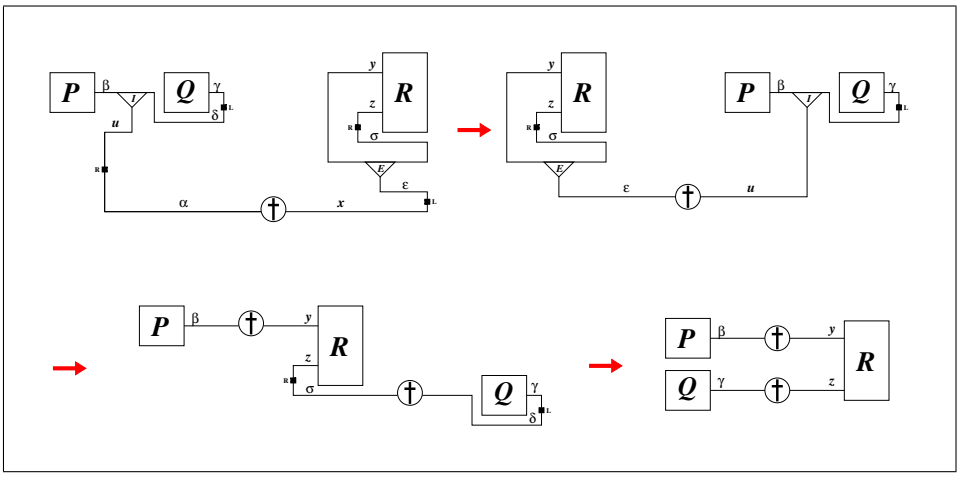



For example in the case of and$_L$, we transform an in-port $z$ into an out-port, by using the negation not$_R$. Then we construct an $\mathcal{E}$-fan using the in-port $y$ and the previously constructed out-port. The result is an out-port, which we convert into an in-port by using the left negation. In the case of and$_R$: we transform the second out-port into an in-port, using not$_L$. As a result we get an in-port which we convert into an out-port by using the right negation. Similar approach is taken when representing left and right disjunction.

**The reduction rules** The diagrammatic view on the new *reduction rules* is as follows:



We show that the decompositions of and$_L$ and and$_R$ by negation and impliciation, as previously presented, are well behaved. This can be checked similarly for or$_L$ and or$_R$.




## 8.9 Notes

**A calculus of non-determinism** The situation with respect to non-determinism in the diagrammatic calculus is rather complex. There are several sources of non-determinism.

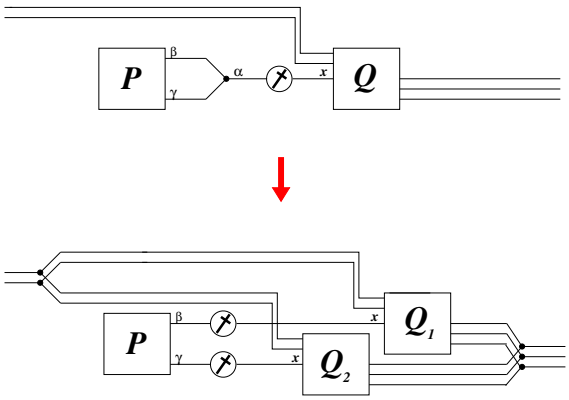
- *The direction of activation.* Sometimes one can choose between left and right-activation of a dagger. Activation rules are a novelty, introduced by the name "commuting cuts" in [UB01a], where it was demonstrated that restrictions used before are not necessary to have strong normalization. The non-determinism reflected by activation rules is a source of non-confluence.

- *Actual boundaries of diagrams.* Diagrams are represented by $\boxed{\text{P}}$. But when working with actual diagrams one may notice another non-determinism. Structural actions define erasure and duplication of diagrams, but the choice which diagram is exactly erased/duplicated is non-deterministic. It can be any subdiagram whose port is bound by a dagger. As in [Rob03] we do not use boxes. They were used in the framework of proof nets for linear logic, where a box is a unit for erasing and duplicating fragments of nets. A solution to partially avoid boxes was proposed in [GAL92].

- *The order of applying the daggers.* Some authors think of this as an angelic non-determinism, but it should be formally proven that it does not lead to non-confluence. A possible way to examine this would be to assume congruence rules (in ${}^*\mathcal{X}$) which do not involve cuts, and then show that using them one can derive congruence rules which involve cuts (the list of congruence rules is assembled in Section 11.2).

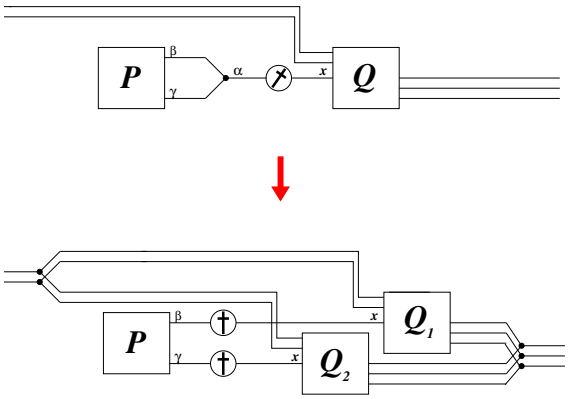
- *The point of attachment of a black hole.* So called black holes are always associated by an already existing diagram, but it is not uniquely determined to which part exactly. It can be attached to an arbitrary subdiagram, which is perhaps too much liberty, as it is difficult to trace the consequences of such non-determinism. Also, during the computation the point of attachment changes when a diagram is erased, as defined by erasure rules. During computation this point has to move, but in general it should be attached to one component (or its subcomponent), and if that component is erased than it should be attached to another diagram, as defined by erasure rules.

**Deactivating daggers after duplication** One of the few restriction in $\mathcal{X}$ calculus (elaborated in [Urb00]), which in some sense violates the perfect symmetry of the sequent calculus, is that active cut can not change its direction of activation at will. This restriction is kept in the calculus because removing it would break strong normalization. However, it has been shown that tis restriction does not restrict the collection of "essential" normal forms. In the framework of ${}^d\mathcal{X}$ we do not have propagation rules. The only moment where we keep the cuts active is in the duplication rules, for example:




The cuts remain activated in the same direction after performing duplica-

tion, and this is also a restriction in ${}^{d}\mathcal{X}$. We assume that if the rule was formulated without this restriction:




it would not break the strong normalization, but we do not study this here. It seems that, using this approach would mean removing the remaining restrictions, and would capture the full symmetry of the sequent calculus. Also, if this was done, there would be no reduction rules whose result includes active cuts. This is in accordance with how we imagine the cuts to be evaluated straight after activation (thus clearly by some structural action).

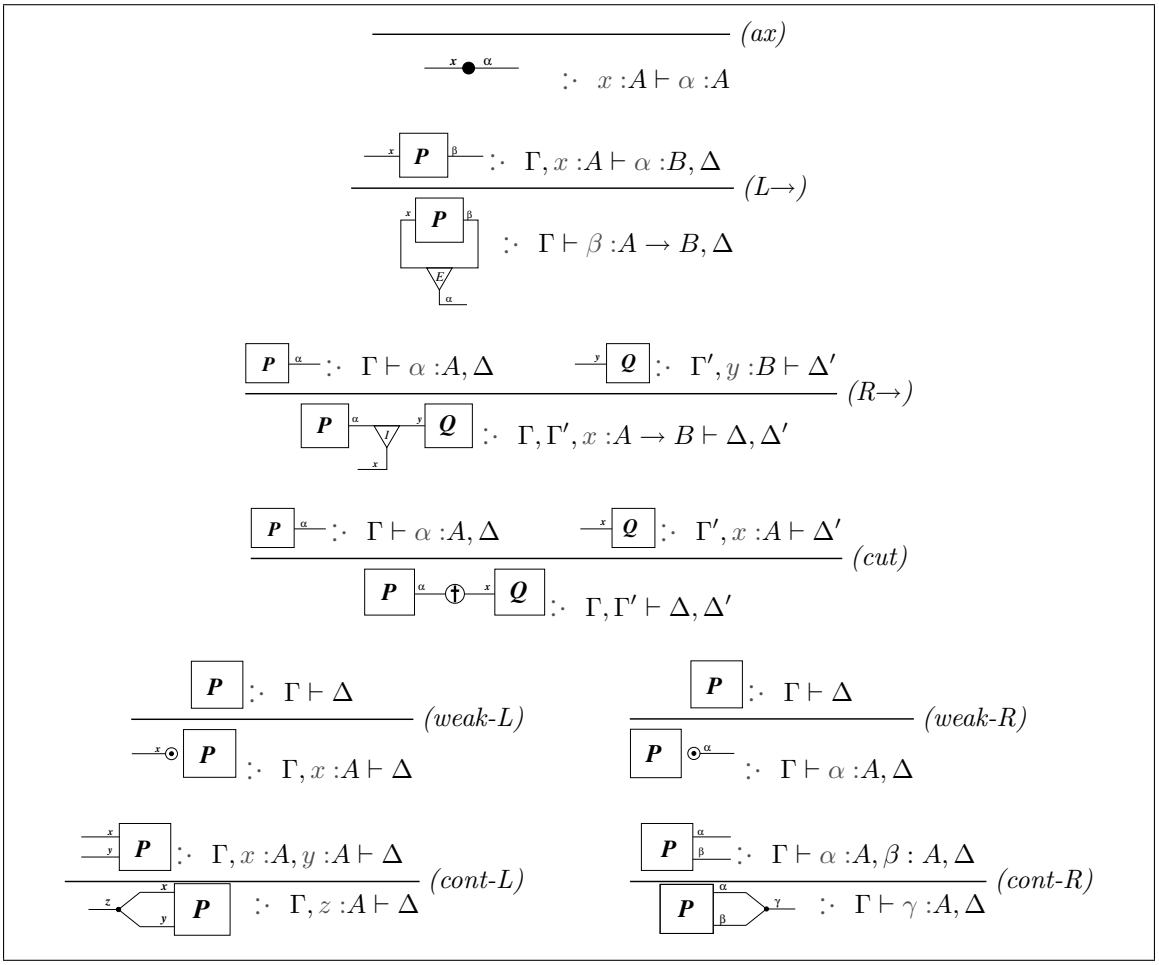



Figure 8.5: The type system for ${}^{d}\mathcal{X}$

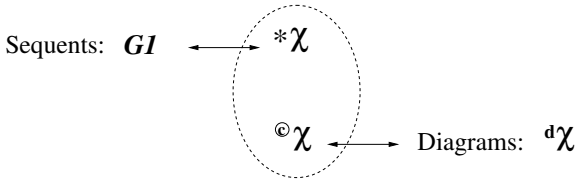
# Part III

# Equivalent Terms in Classical Computation

# Chapter 9

# Introduction

The third part of this thesis presents another calculus, called $^{\copyright}\mathcal{X}$, which is created as a result of studying the relation between the $^{*}\mathcal{X}$ calculus, presented in Part I, and the $^{d}\mathcal{X}$ calculus, presented in Part II.




The $^{\copyright}\mathcal{X}$ calculus is a one-dimensional counterpart of the diagrammatic calculus. It is obtained from $^{*}\mathcal{X}$ by introducing a congruence relation on its terms. The congruence rules define which syntactically different $^{*}\mathcal{X}$-terms should be considered the same. Since $^{*}\mathcal{X}$ codes the sequent calculus terms, this means that we define *which* syntactically different classical sequent proofs should be considered the same.

We were researching the permutation of independent inference rules in the framework of $^{*}\mathcal{X}$-terms instead of sequent proofs. Later on, an analogy with another work has been discovered. Namely, the study we present in this part is related to the results presented by Robinson in the framework of proof nets for classical logic [Rob03], where, in Section 5, the problem of permutation of inference rules was addressed. It has been shown that "two proofs give the same net if and only if each one can be derived from the other by a sequence of trivial rule permutations". But the details of how the transposition of rules should be managed and when the rules can be transposed were not presented.

In Robinson's opinion "...there are numerous (largely irrelevant) subcases depending on exactly which bits of proofs different formulae come from, and it would be tedious and unilluminating to provide the definition in full detail". Instead some illustrative examples are given.

Our viewpoint is different. On the one hand, this task is important because it is relevant for the next step in our study, how dynamics of diagrams relates to the dynamics of ${}^*\mathcal{X}$-terms (proofs).[1] We will try to show that there is a strong relation between the dynamics of diagrams and the dynamics of ${}^*\mathcal{X}$-terms, in a sense that diagram reductions correspond to ${}^*\mathcal{X}$-reductions modulo restructuring which corresponds to permuting independent rules. To show this we need to know when and how we can perform the restructuring of terms.

On the other hand, this task was not too complicated. Indeed, if one is to present these rules using sequent calculus, it would be very difficult. But since we already have both calculi at hand, the one-dimensional ${}^*\mathcal{X}$ which codes sequent proofs, and the diagrammatic ${}^d\mathcal{X}$, the idea to collect all terms which are represented by the same diagram came naturally as a consequence of studying the relation between these two calculi.

In that sense the third part of this thesis also addresses the very old problem of proof identity, which is caused by the lack of understanding of the essence of a proof, which is in turn caused by the bureaucracy involved in the syntactic presentation of proofs.

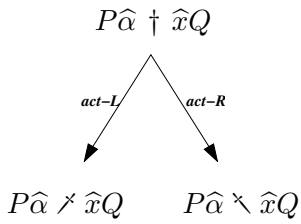
[1] We are mainly concerned by the dynamics of the calculi. We use the results presented by Robinson which deal with static aspects of proof nets, and their relation with actual proofs.

# Chapter 10

# One-dimensional vs. two-dimensional computation

Having both calculi at hand, the one-dimensional one of ${}^*\mathcal{X}$ and the diagrammatic of ${}^d\mathcal{X}$, we may compare them to better understand the differences as well as the similarities.

One of the principal differences is illustrated by the following situation. If we want to reduce the term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$, where the names $\alpha$ and $x$ are somewere deep inside in the structure of $P$ and $Q$, respectively, then we have to do *cut-propagations*. To propagate a cut means to push a cut through the structure of terms, in order to reach one of the names involved, $\alpha$ or $x$. This process is similar to the explicit substitution propagation, as defined in the $\lambda\mathsf{x}$ calculus [BR95], and even more similar to the one described for $\lambda\mathsf{lxr}$ [KL05, KL07] where weakening and contraction are explicit. Unlike in these intuitionistic calculi, propagations in ${}^*\mathcal{X}$ can be done in both directions, which is shown by bending the cut:

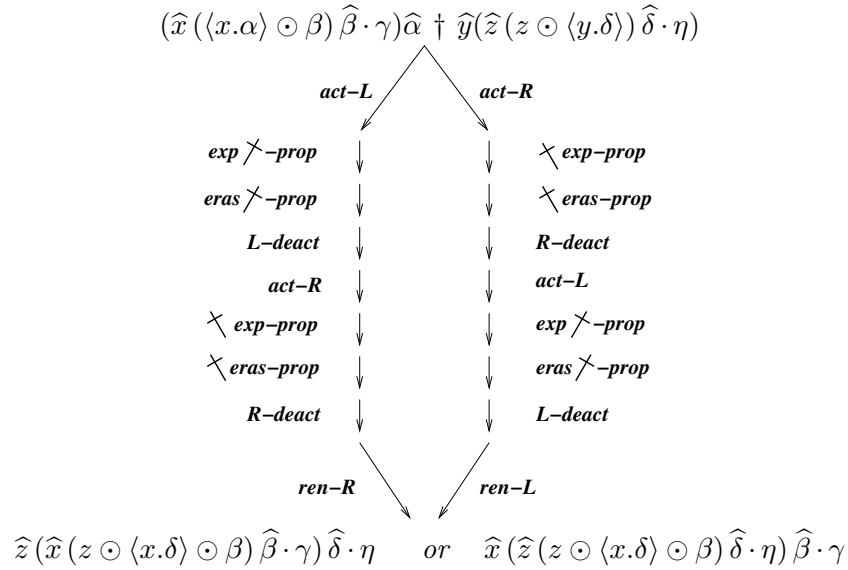



In the first case, the module $\widehat{\alpha} \nearrow \widehat{x}Q$ is going to be propagated through the structure of $P$ (if $P$ does not introduce $\alpha$), and the same stands for the module $P\widehat{\alpha} \nwarrow \widehat{x}$, which can be propagated through $S$.

If there is no need to do cut-propagation, that is, if $\alpha$ and $x$ are already

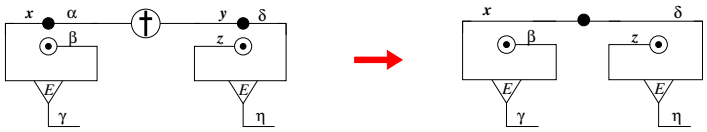
introduced (at the top level) by $P$ and $Q$, there is no essential difference in one-dimensional and two-dimensional computation, i.e., between ${}^{*}\mathcal{X}$ and ${}^{d}\mathcal{X}$. In other words, if we neglect propagation rules, the core of the computation engine, i.e., activation, deactivation, logical and structural rules, works the same for diagrams and terms.

**Example 64** *The following example illustrates two facts. Firstly, it shows that the number of reduction steps is smaller in the case of diagrammatic computing. Secondly, it shows the discrepancy between a one-dimensional and a two-dimensional syntax. Indeed, there are two different resulting terms in the one-dimensional case and only one in the two-dimensional case. We will argue that two resulting one-dimensional terms are syntactically different, but essentially the same, and that both correspond to the unique resulting diagram.*




There are nine reduction steps between the starting term and each of the two resulting terms. Two branches are possible due to non-determinism in the reduction procedure. For us, in the framework of the ${}^{d}\mathcal{X}$ calculus, the two resulting terms will be essentially the same, since they represent sequent proofs which differ only in the trivial permutation of inference rules.

**Example 65** *Diagrammatically, the same computation is done in one step, as presented by the following reduction:*

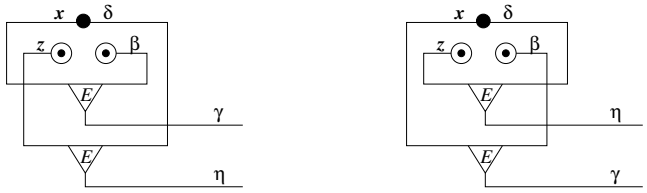



This is because free *ports* in diagrams are always directly accessible, which is not the case with free *names* in terms, where it is often necessary to do propagations in order to reach certain free names.

In the one-dimensional example above, the resulting terms are clearly syntactically different. But it also stands that they correspond to the same diagram, which suggests that they should be seen as equivalent. Thus we may define a congruence rule

$$\widehat{z}\,(\widehat{x}\,(z \odot \langle x.\delta\rangle \odot \beta)\,\widehat{\beta}\cdot\gamma)\,\widehat{\delta}\cdot\eta \;\equiv\; \widehat{x}\,(\widehat{z}\,(z \odot \langle x.\delta\rangle \odot \beta)\,\widehat{\delta}\cdot\eta)\,\widehat{\beta}\cdot\gamma$$

Notice that the conditions $\gamma \neq \delta$ and $\eta \neq \beta$ are fulfilled because of Barendregt's convention. However, if one is interested in details, the diagrammatic representation of the two terms can be given in two styles:




The above story, which relates terms and diagrams, refers in fact to a sequent calculus proof which is presented in what follows. Notice that the cut formulas (in oval boxes) are introduced at the level of axioms (marked with a star):

$$\dfrac{\dfrac{\dfrac{}{A \vdash A^*}\,(ax)}{A \vdash A, B}\,(wr)}{\vdash \boxed{\mathrm{A}}, A \to B}\,(\to R) \qquad \dfrac{\dfrac{\dfrac{}{A^* \vdash A}\,(ax)}{A, C \vdash A}\,(wl)}{\boxed{\mathrm{A}} \vdash C \to A}\,(\to R)}{\vdash A \to B,\ C \to A}\,(cut)$$

This proof normalizes to either

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}(ax)}{A, C \vdash A, B}(wr, wl)}{C \vdash A \to B,\ A}(\to R^1)}{\vdash A \to B,\ C \to A}(\to R^2) \qquad \text{or} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}(ax)}{A, C \vdash A, B}(wr, wl)}{A \vdash B,\ C \to A}(\to R^2)}{\vdash A \to B,\ C \to A}(\to R^1)$$

where the two resulting proofs differ in the order of applying the two independent $(\to R)$ inference rules.[1]

It is quite common in ${}^*\mathcal{X}$ (and the same stands for sequent calculus proofs) that some terms are syntactically different, whereas there is no essential difference between them. That is why we have a large number of congruence rules in the system. As said, the congruence relation is introduced to come closer to a diagrammatic calculus. It is also significant because we present a collection of congruence rules, so that we can actually see *which* terms should be identified. As ${}^*\mathcal{X}$-terms directly code sequent proofs, this also means identifying sequent calculus proofs.

[1]The two weakening rules can also be permuted, but we do not address them in this example.

# Chapter 11

# Computing and equivalent terms: the $^{©}\mathcal{X}$ calculus

In this chapter we present the $^{©}\mathcal{X}$ calculus. It is obtained from the $^{*}\mathcal{X}$ calculus by introducing congruence relation on its terms. They have the same syntax, but there are differences at the level of reduction rules. Although closely related, the two calculi differ in that the reduction relation in $^{©}\mathcal{X}$ is defined modulo congruence rules. One of the consequences of this approach is that propagation rules are superfluous, which is a step in a positive direction, considering that we wish $^{©}\mathcal{X}$ to mirror the diagrammatic calculus.

## 11.1 The syntax

The syntax of $^{©}\mathcal{X}$ calculus is the same as that of $^{*}\mathcal{X}$ calculus.

| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $\langle x.\alpha \rangle$ | *capsule* |
| | $\mid$ | $\widehat{x}\, P\, \widehat{\beta} \cdot \alpha$ | *exporter* |
| | $\mid$ | $P\, \widehat{\alpha}\ [x]\ \widehat{y}\, Q$ | *importer* |
| | $\mid$ | $P \widehat{\alpha}\ \dagger\ \widehat{x} Q$ | *cut* |
| | $\mid$ | $x \odot P$ | *left-eraser* |
| | $\mid$ | $P \odot \alpha$ | *right-eraser* |
| | $\mid$ | $z{<}^{\widehat{x}}_{\widehat{y}}\langle P]$ | *left-duplicator* |
| | $\mid$ | $[P\rangle^{\widehat{\alpha}}_{\widehat{\beta}}{>}\gamma$ | *right-duplicator* |

Figure 11.1: The syntax of $^{©}\mathcal{X}$

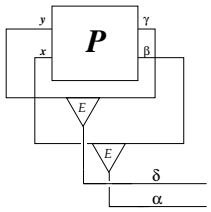
## 11.2 The congruence relation

In this section we define a congruence relation on terms, denoted by $\equiv$. It is defined in what follows by a list of equations. The relation $\equiv$ is a reflexive, symmetric and transitive relation closed under any context [Ter03], which contains the equations given in this section. The motivation for introducing it into the system is to bring the one-dimensional syntax closer to the diagrammatic calculus.

Congruence rules are listed below and they induce the congruence relation on terms. They define which syntactically different terms have the same diagrammatic represention, that is, which syntactically different terms should be considered the same. Every rule is associated with the corresponding diagram. A name is assigned to every congruence rule, and they are presented in the form: $name : P \equiv Q$.

---

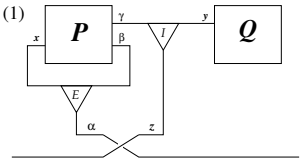
exp-exp

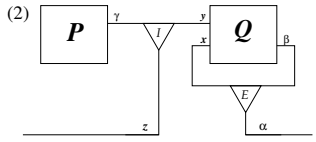



$ee \;:\; \widehat{y}\,(\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\cdot\delta \equiv \widehat{x}\,(\widehat{y}\,P\,\widehat{\gamma}\cdot\delta)\,\widehat{\beta}\cdot\alpha$

---

exp-imp



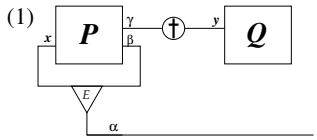



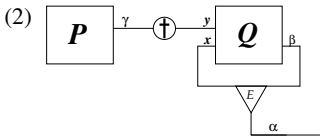

$ei1 \;:\; \widehat{x}\,(P\,\widehat{\gamma}\;[z]\;\widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\;[z]\;\widehat{y}\,Q$ with $x, \beta \in N(P)$

$ei2 \;:\; \widehat{x}\,(P\,\widehat{\gamma}\;[z]\;\widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; P\,\widehat{\gamma}\;[z]\;\widehat{y}\,(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)$ with $x, \beta \in N(Q)$

---

exp-cut



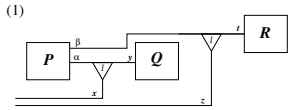



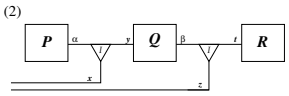

$ec1 \;:\; \widehat{x}\,(P\widehat{\gamma} \dagger \widehat{y}Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\gamma} \dagger \widehat{y}Q$ with $x, \beta \in N(P)$

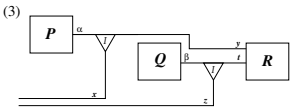
$ec2 \;:\; \widehat{x}\,(P\widehat{\gamma} \dagger \widehat{y}Q)\,\widehat{\beta}\cdot\alpha \;\equiv\; P\widehat{\gamma} \dagger \widehat{y}(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)$ with $x, \beta \in N(Q)$

---

imp-imp



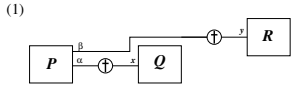

$ii1 \;:\; (P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q)\,\widehat{\beta}\;[z]\;\widehat{t}\,R \;\equiv\; (P\,\widehat{\beta}\;[z]\;\widehat{t}\,R)\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q$ with $\alpha, \beta \in N(P)$

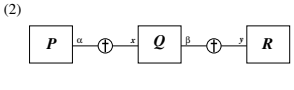
$ii2 \;:\; (P\,\widehat{\alpha}\;[x]\;\widehat{y}\,Q)\,\widehat{\beta}\;[z]\;\widehat{t}\,R \;\equiv\; P\,\widehat{\alpha}\;[x]\;\widehat{y}\,(Q\,\widehat{\beta}\;[z]\;\widehat{t}\,R)$ with $y, \beta \in N(Q)$

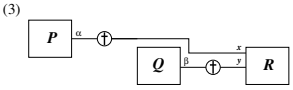
$ii3 \;:\; (Q\,\widehat{\beta}\;[z]\;\widehat{t}\,R) \;\equiv\; Q\,\widehat{\beta}\;[z]\;\widehat{t}\,(P\,\widehat{\alpha}\;[x]\;\widehat{y}\,R)$ with $y, t \in N(R)$

---

cut-cut



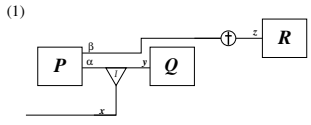

$cc1 \;:\; (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R \;\equiv\; (P\widehat{\beta} \dagger \widehat{y}R)\widehat{\alpha} \dagger \widehat{x}Q$ with $\alpha, \beta \in N(P)$

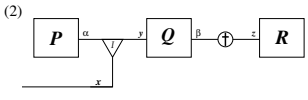
$cc2 \;:\; (P\widehat{\alpha} \dagger \widehat{x}Q)\widehat{\beta} \dagger \widehat{y}R \;\equiv\; P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R)$ with $x, \beta \in N(Q)$

$cc3 \;:\; P\widehat{\alpha} \dagger \widehat{x}(Q\widehat{\beta} \dagger \widehat{y}R) \;\equiv\; Q\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}R)$ with $x, y \in N(R)$

---

cut-imp



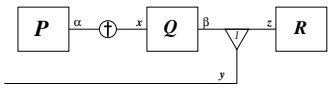



$ci1$ : $(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\widehat{\beta}\,\dagger\,\widehat{z}R \;\equiv\; (P\widehat{\beta}\,\dagger\,\widehat{z}R)\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q$ with $\alpha, \beta \in N(P)$

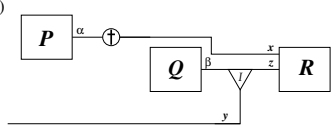
$ci2$ : $(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\widehat{\beta}\,\dagger\,\widehat{z}R \;\equiv\; P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\widehat{\beta}\,\dagger\,\widehat{z}R)$ with $y, \beta \in N(Q)$

---

(3)




(4)



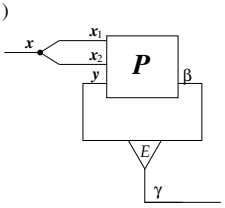

$ci3$ : $P\widehat{\alpha}\,\dagger\,\widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\equiv\; (P\widehat{\alpha}\,\dagger\,\widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,R$ with $x, \beta \in N(Q)$

$ci4$ : $P\widehat{\alpha}\,\dagger\,\widehat{x}(Q\,\widehat{\beta}\,[y]\,\widehat{z}\,R) \;\equiv\; Q\,\widehat{\beta}\,[y]\,\widehat{z}\,(P\widehat{\alpha}\,\dagger\,\widehat{x}R)$ with $x, z \in N(R)$

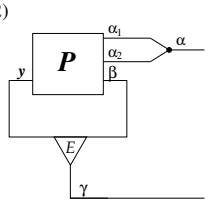
---

exp-cont

(1)




(2)



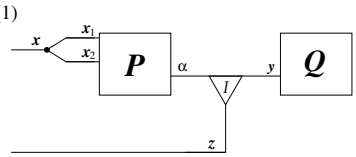

$ec_t1$ : $x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle\widehat{y}\,P\,\widehat{\beta}\cdot\gamma] \;\equiv\; \widehat{y}\,(x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P])\,\widehat{\beta}\cdot\gamma$ with $y \neq x$

$ec_t2$ : $[\widehat{y}\,P\,\widehat{\beta}\cdot\gamma\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}{>}\alpha \;\equiv\; \widehat{y}\,([P\rangle\frac{\widehat{\alpha_1}}{\widehat{\alpha_2}}{>}\alpha)\,\widehat{\beta}\cdot\gamma$ with $\alpha \neq \beta,\;\; \alpha_1, \alpha_2 \neq \gamma$

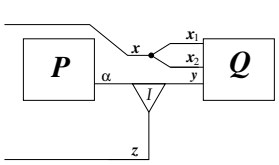
---

imp-cont

(1)




(2)

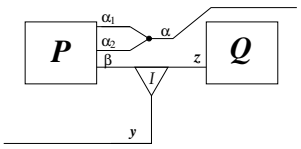



$ic_t1$ : $x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P\,\widehat{\alpha}\,[z]\,\widehat{y}\,Q] \;\equiv\; (x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P])\,\widehat{\alpha}\,[z]\,\widehat{y}\,Q$ with $x_1, x_2 \in N(P)$

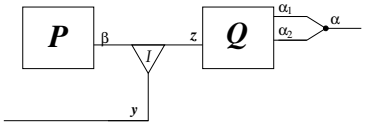
$ic_t2$ : $x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P\,\widehat{\alpha}\,[z]\,\widehat{y}\,Q] \;\equiv\; P\,\widehat{\alpha}\,[z]\,\widehat{y}\,(x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle Q])$ with $x_1, x_2 \in N(Q),\;\; y \neq x$

---

(3)




(4)



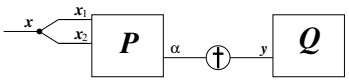

$ic_t3$ : $[P\,\widehat{\beta}\,[z]\,\widehat{y}\,Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha \;\equiv\; ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\,\widehat{\beta}\,[z]\,\widehat{y}\,Q$ with $\alpha_1, \alpha_2 \in N(P),\ \alpha \neq \beta$

$ic_t4$ : $[P\,\widehat{\beta}\,[z]\,\widehat{y}\,Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha \;\equiv\; P\,\widehat{\beta}\,[z]\,\widehat{y}\,([Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)$ with $\alpha_1, \alpha_2 \in N(Q)$

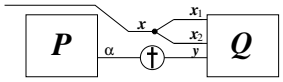
---

cut-cont

(1)




(2)

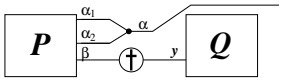



$cc_t1$ : $x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\widehat{\alpha}\,\dagger\,\widehat{y}Q] \;\equiv\; (x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P])\widehat{\alpha}\,\dagger\,\widehat{y}Q$ with $x_1, x_2 \in N(P)$

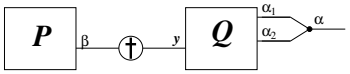
$cc_t2$ : $x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle P\widehat{\alpha}\,\dagger\,\widehat{y}Q] \;\equiv\; P\widehat{\alpha}\,\dagger\,\widehat{y}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}\langle Q])$ with $x_1, x_2 \in N(Q),\ y \neq x$

---

(3)




(4)



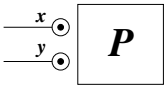

$cc_t3$ : $[P\widehat{\beta}\,\dagger\,\widehat{y}Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha \;\equiv\; ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\beta}\,\dagger\,\widehat{y}Q$ with $\alpha_1, \alpha_2 \in N(P),\ \alpha \neq \beta$

$cc_t4$ : $[P\widehat{\beta}\,\dagger\,\widehat{y}Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha \;\equiv\; P\widehat{\beta}\,\dagger\,\widehat{y}([Q\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)$ with $\alpha_1, \alpha_2 \in N(Q)$

---

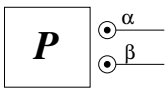
weak-weak

(1)




(2)


(3)




$ww1$ : $y \odot (x \odot P) \;\equiv x \odot (y \odot P)$

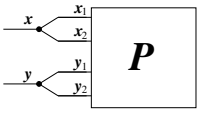
$ww2$ : $x \odot (P \odot \alpha) \;\equiv\; (x \odot P) \odot \alpha$

$ww3$ : $(P \odot \alpha) \odot \beta \;\equiv\; (P \odot \beta) \odot \alpha$

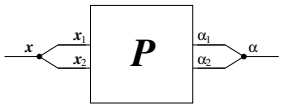
---

cont-cont

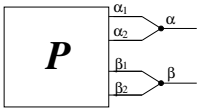
(1)




(2)




(3)




$c_t c_t 1$ : $y<\widehat{\frac{y_1}{y_2}}\langle x<\widehat{\frac{x_1}{x_2}}\langle P]] \equiv x<\widehat{\frac{x_1}{x_2}}\langle y<\widehat{\frac{y_1}{y_2}}\langle P]]$ with $y \notin \{x_1, x_2\}$ and $x \notin \{y_1, y_2\}$

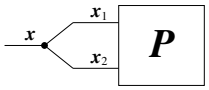
$c_t c_t 2$ : $[x<\widehat{\frac{x_1}{x_2}}\langle P]\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\alpha \equiv x<\widehat{\frac{x_1}{x_2}}\langle[P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\alpha]$

$c_t c_t 3$ : $[[P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\alpha\rangle\widehat{\frac{\beta_1}{\beta_2}}>\beta \equiv [[P\rangle\widehat{\frac{\beta_1}{\beta_2}}>\beta\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\alpha$ with $\beta \notin \{\alpha_1, \alpha_2\}$ and $\alpha \notin \{\beta_1, \beta_2\}$

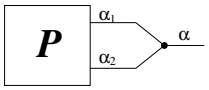
---

cont-comm

(1)




(2)



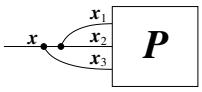

$c_t cm1$ : $x<\widehat{\frac{x_1}{x_2}}\langle P] \equiv x<\widehat{\frac{x_2}{x_1}}\langle P]$

$c_t cm2$ : $[P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\alpha \equiv [P\rangle\widehat{\frac{\alpha_2}{\alpha_1}}>\alpha$

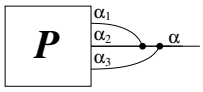
---

cont-assoc

(1)




(2)



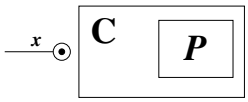

$c_t as1$ : $z<\widehat{\frac{y}{x_3}}\langle y<\widehat{\frac{x_1}{x_2}}\langle P]] \equiv z<\widehat{\frac{x_1}{y}}\langle y<\widehat{\frac{x_2}{x_3}}\langle P]]$

$c_t as2$ : $[[P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\beta\rangle\widehat{\frac{\beta}{\alpha_3}}>\gamma \equiv [[P\rangle\widehat{\frac{\alpha_2}{\alpha_3}}>\beta\rangle\widehat{\frac{\alpha_1}{\beta}}>\gamma$

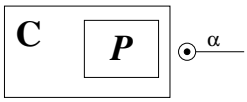
---

weak-general

(1)




(2)




$wg1$ : $x \odot (C\{P\}) \equiv C\{x \odot P\}$ $x \notin N(C\{P\})$

$wg2$ : $(C\{P\}) \odot \alpha \equiv C\{P \odot \alpha\}$ $\alpha \notin N(C\{P\})$

---

The rules are divided into eighteen groups, making all possible combinations among the terms introduced by the syntax. Each group usually has two or three rules.

The relation $\equiv$ induces congruence classes on terms. We use $cl(P)$ to denote the congruence class of a term $P$ with respect to a relation $\equiv$. Notice that each congruence class has finitely many terms. Since it is obvious that the two congruent terms have the same size, it is not difficult to prove that there is finitely many possibilities to pick up a representative of a class.

Congruence rules satisfy some standard properties such as preservation of linearity, preservation of free names and preservation of types.

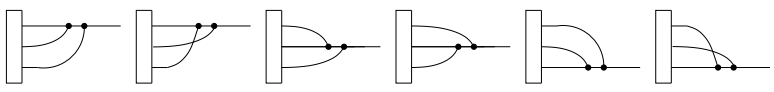
While the majority of the congruence rules only describe the different views we may have of the same diagram, namely from (exp-exp) to (cont-cont), the last three congruence rules are of slightly different nature. Therefore they are separated from the rest by a thicker line. In what follows we will describe their intended meaning.

<u>*cont-comm*</u>: In the *cont-comm* group, the rules say that we can commute parts under a contraction. At the level of diagrams we could have two representatives for $\overset{c_t cm1}{\equiv}$, say:


but since the above-below order does not play a role, we will consider the two diagrams to be the same. This is one of the points which suggests that there is a flavor of a third dimension in our diagrammatic model.

<u>*cont-assoc*</u>: The rules in this group speak about associativity under contraction. At the diagrammatic level, for example for $\overset{c_t as1}{\equiv}$, there are two possible representatives, but we implicitly decide to see them as one. These are


which means that for us order of successive forks is not relevant. This practically means that we are are considering forks of higher arity, but formally we will stay at arity 2. Finally, by combining *cont-comm* and *cont-assoc* we get that the following fork combinations are viewed as the same:



*weak-general*: The last congruence rule is general, in the sense that $C$ can be any context. It says that the weakening (black-hole) is related to a whole component, i.e., we can choose to which exactly part a component to attach it (this approach was taken in [Rob03]). The component is like one magnet which, in its total, attracts the black-hole. It is also possible to introduce the the explicit link (dotted line), which connects a black hole with a component it refers to. But, as said, this is non-deterministic, since we may attach it to any part of the component, as illustrated.[1]


## 11.3 Restructuring terms

It has been shown in [Rob03] that two sequent proofs induce the same proof net if and only if one can be obtained from the other by a sequence of transpositions of independent rules. In this thesis we proceed further to study the *dynamic* level of this relation. In what follows we show that the congruence rules allow us to perform restructuring of $^{*}\mathcal{X}$-terms (i.e. sequent proofs), so that the cut-names are brought to the top level. This will allow us to to show (in terms of relating $^{d}\mathcal{X}$ and ©$\mathcal{X}$-reductions), that a reduction steps of diagrams correspond to reduction steps performed on sequent proofs, modulo restructuring by permuting the independent rules (see Chapter 11.6).

When we analyze the computation in the diagrammatic calculus, we notice that free ports are always accessible. This is due to the fact that in the two-dimensional space we do not have sequentialization, which exists in the one-dimensional models, and which in the case of the $^{*}\mathcal{X}$ calculus originates from the sequent calculus. In the sequent calculus this shows at the level of dynamics - it is needed to do step-by-step cut propagation in order to reach the cut formula, and more importantly at the static level - it allows only one inference rule to be applied at a time. All these features are captured by the $^{*}\mathcal{X}$ calculus.

[1] There is always a pre-existing component. This comes from the sequent calculus, where the empty sequent is not provable.

Take for example an arbitrary ${}^*\mathcal{X}$-term of the form:

$$P\widehat{\alpha} \dagger \widehat{x}Q$$

Here, the name $\alpha$ might not be directly accessible (that is, $\alpha$ is maybe not introduced by $P$). Furthermore, this might hold for both names involved in the cut, $\alpha$ and $x$, which may occur deeply inside of their corresponding terms. We prove that it is possible to transform the above term - using only congruence rules defined in Section 11.2 - to the form

$$C\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q^{x}\}$$

where $C$ is a context, and $\alpha$ and $x$ are principal names of $P^{\alpha} \preccurlyeq P$ and $Q^{x} \preccurlyeq Q$, respectively. In other words we can always pick at least one representative of a congruence class $cl(P\widehat{\alpha} \dagger \widehat{x}Q)$, which allows us to continue the reduction process.

Recall that $\preccurlyeq$ stands for a subterm relation (Definition 11). Recall also that $cl(P)$ denotes the congruence class of a term $P$ with respect to a congruence relation $\equiv$.

In what follows we first formulate two lemmas which focus only on one cut-name (either $x$ or $\alpha$) at a time. Then we give the main theorem, which refers to both names. In proving the results we will use the transitivity of $\preccurlyeq$ relation: If $P \preccurlyeq Q$ and $Q \preccurlyeq R$ then $P \preccurlyeq R$.

**Lemma 66 (Left-propagation lemma)** *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$, there exists a context $C$ and a term $P^{\alpha}$, where $\alpha$ is a principal name for $P^{\alpha}$ and $P^{\alpha} \preccurlyeq P$, such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}$$

**Proof:** By induction on the structure of $P$ and case analysis.

- The base case is when $\alpha$ is a principal name for $P$.
  Then we have $P^{\alpha} = P$ and $C = \{\ \}$.
- Assume that the property holds for the immediate subterms of $P$.
- The possible cases for $P$ are

| | | |
|---|---|---|
| 1. | $P = \widehat{y}\, M\, \widehat{\beta} \cdot \gamma,$ | $\gamma \neq \alpha$ |
| 2. | $P = M\, \widehat{\beta}\ [y]\ \widehat{z}\, N$ | |
| 3. | $P = M\widehat{\beta} \dagger \widehat{y}N$ | |
| 4.&5. | $P = x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle M]$ and $P = [M\rangle\frac{\widehat{\beta_1}}{\widehat{\beta_2}}{>}\beta,$ | $\beta \neq \alpha$ |
| 6.&7. | $P = x \odot M$ and $P = M \odot \alpha,$ | $\beta \neq \alpha$ |

We analyze all the cases:

1. $(\widehat{y}\,M\,\widehat{\beta}\cdot\gamma)\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{ec1}{\equiv}$ $\widehat{y}\,(M\widehat{\alpha}\dagger\widehat{x}Q)\,\widehat{\beta}\cdot\gamma$
$\overset{i.h.}{\equiv}$ $\widehat{y}\,(C'\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})\,\widehat{\beta}\cdot\gamma$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = \widehat{y}\,(C'\{\ \})\,\widehat{\beta}\cdot\gamma$

2. $(a)$ Case $\alpha\in M$.
$(M\,\widehat{\beta}\,[y]\,\widehat{z}\,N)\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{ci1}{\equiv}$ $(M\widehat{\alpha}\dagger\widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\overset{i.h.}{\equiv}$ $(C'\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = (C'\{\ \})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$

$(b)$ Case $\alpha\in N$.
$(M\,\widehat{\beta}\,[y]\,\widehat{z}\,N)\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{ci2}{\equiv}$ $(M\widehat{\alpha}\dagger\widehat{x}Q)\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\overset{i.h.}{\equiv}$ $(C'\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})\,\widehat{\beta}\,[y]\,\widehat{z}\,N$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = M\,\widehat{\beta}\,[y]\,\widehat{z}\,(C'\{\ \})$

3. $(a)$ Case $\alpha\in M$.
$(M\widehat{\beta}\dagger\widehat{y}N)\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{cc1}{\equiv}$ $(M\widehat{\alpha}\dagger\widehat{x}Q)\widehat{\beta}\dagger\widehat{y}N$
$\overset{i.h.}{\equiv}$ $(C'\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})\widehat{\beta}\dagger\widehat{y}N$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = (C'\{\ \})\widehat{\beta}\dagger\widehat{y}N$

$(b)$ Case $\alpha\in N$.
$(M\widehat{\beta}\dagger\widehat{y}N)\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{cc2}{\equiv}$ $M\widehat{\beta}\dagger\widehat{y}(N\widehat{\alpha}\dagger\widehat{x}Q)$
$\overset{i.h.}{\equiv}$ $M\widehat{\beta}\dagger\widehat{y}(C'\{N^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})$
$\triangleq$ $C\{N^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = M\widehat{\beta}\dagger\widehat{y}(C'\{\ \})$

4. $(y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle M])\widehat{\alpha}\dagger\widehat{x}Q$
$\overset{cc_t1}{\equiv}$ $y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle(M\widehat{\alpha}\dagger\widehat{x}Q)]$
$\overset{i.h.}{\equiv}$ $y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle(C'\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\})]$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha}\dagger\widehat{x}Q\}$,
with $C\{\ \} = y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle C'\{\ \}]$

5. $([M\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}>\beta)\widehat{\alpha} \dagger \widehat{x}Q$
$\overset{cc_t3}{\equiv}$ $[(M\widehat{\alpha} \dagger \widehat{x}Q)\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}>b$
$\overset{i.h.}{\equiv}$ $[(C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}>\beta$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}$,
with $C\{\ \} = [C'\{\ \}\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}>\beta$

6. $(y \odot M)\widehat{\alpha} \dagger \widehat{x}Q$
$\overset{wg1}{\equiv}$ $y \odot (M\widehat{\alpha} \dagger \widehat{x}Q)$
$\overset{i.h.}{\equiv}$ $y \odot (C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\})$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}$,
with $C\{\ \} = y \odot (C'\{\ \})$

7. $(M \odot \beta)\widehat{\alpha} \dagger \widehat{x}Q$
$\overset{wg2}{\equiv}$ $(M\widehat{\alpha} \dagger \widehat{x}Q) \odot \beta$
$\overset{i.h.}{\equiv}$ $(C'\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}) \odot \beta$
$\triangleq$ $C\{M^{\alpha}\widehat{\alpha} \dagger \widehat{x}Q\}$,
with $C\{\ \} = (C'\{\ \}) \odot \beta$

Thus we are done with the first lemma. □

**Lemma 67 (Right-propagation lemma)** *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$, there exists a context $C$ and a term $Q^x$, whose principal name is $x$ and $Q^x \preccurlyeq Q$, such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P\widehat{\alpha} \dagger \widehat{x}Q^x\}$$

**Proof:** By induction on the structure of $Q$ and case analysis.

- Case $x$ is a principal name for $Q$. Then $Q^x = Q$ and $C = C\{\ \}$.
- We assume that the property holds for the immediate subterms of $Q$.
- The possible cases for $Q$ are

| | | |
|---|---|---|
| 1. | $Q = \widehat{y}\, M\, \widehat{\beta} \cdot \gamma$ | |
| 2. | $Q = M\, \widehat{\beta}\, [y]\, \widehat{z}\, N$ | |
| 3. | $Q = M\widehat{\beta} \dagger \widehat{y}N$, | $y \neq x$ |
| 4.&5. | $Q = x<^{\widehat{x_1}}_{\widehat{x_2}}\langle M]$ and $P = [M\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}>\beta$, | $y \neq x$ |
| 6.&7. | $Q = x \odot M$ and $P = M \odot \alpha$, | $y \neq x$ |

We analyze these seven cases:

1. $P\widehat{\alpha} \dagger \widehat{x}(\widehat{y}\, M\, \widehat{\beta} \cdot \gamma)$
$\overset{ec2}{\equiv}$ $\widehat{y}\,(P\widehat{\alpha} \dagger \widehat{x}M)\,\widehat{\beta} \cdot \gamma$
$\overset{i.h.}{\equiv}$ $\widehat{y}\,(C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\})\,\widehat{\beta} \cdot \gamma$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}$,
with $C\{\ \} = \widehat{y}\,(C'\{\ \})\,\widehat{\beta} \cdot \gamma$

2. $(a)$ Case $x \in M$.
$P\widehat{\alpha} \dagger \widehat{x}(M\,\widehat{\beta}\ [y]\ \widehat{z}\, N)$
$\overset{ic3}{\equiv}$ $(P\widehat{\alpha} \dagger \widehat{x}M)\,\widehat{\beta}\ [y]\ \widehat{z}\, N$
$\overset{i.h.}{\equiv}$ $(C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\})\,\widehat{\beta}\ [y]\ \widehat{z}\, N$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}$,
with $C\{\ \} = (C'\{\ \})\,\widehat{\beta}\ [y]\ \widehat{c}N$

$(b)$ Case $x \in N$.
$P\widehat{\alpha} \dagger \widehat{x}(M\,\widehat{\beta}\ [y]\ \widehat{z}\, N)$
$\overset{ic4}{\equiv}$ $M\,\widehat{\beta}\ [y]\ \widehat{z}\,(P\widehat{\alpha} \dagger \widehat{x}N)$
$\overset{i.h.}{\equiv}$ $M\,\widehat{\beta}\ [y]\ \widehat{z}\,(C'\{P\widehat{\alpha} \dagger \widehat{x}N^x\})$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}N^x\}$,
with $C\{\ \} = M\,\widehat{\beta}\ [y]\ \widehat{c}\,(C'\{\ \})$

3. $(a)$ Case $x \in M$.
$P\widehat{\alpha} \dagger \widehat{x}(M\widehat{\beta} \dagger \widehat{y}N)$
$\overset{cc2}{\equiv}$ $(P\widehat{\alpha} \dagger \widehat{x}M)\widehat{\beta} \dagger \widehat{y}N$
$\overset{i.h.}{\equiv}$ $(C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\})\widehat{\beta} \dagger \widehat{y}N$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}$,
with $C\{\ \} = (C'\{\ \})\widehat{\beta} \dagger \widehat{y}N$

$(b)$ Case $x \in N$.
$P\widehat{\alpha} \dagger \widehat{x}(M\widehat{\beta} \dagger \widehat{y}N)$
$\overset{cc3}{\equiv}$ $M\widehat{\beta} \dagger \widehat{y}(P\widehat{\alpha} \dagger \widehat{x}N)$
$\overset{i.h.}{\equiv}$ $M\widehat{\beta} \dagger \widehat{y}(C'\{P\widehat{\alpha} \dagger \widehat{x}N^x\})$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}N^x\}$,
with $C\{\ \} = M\widehat{\beta} \dagger \widehat{y}(C'\{\ \})$

4. $P\widehat{\alpha} \dagger \widehat{x}(y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle M])$
$\overset{cc_t2}{\equiv}$ $y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle (P\widehat{\alpha} \dagger \widehat{x}M)]$
$\overset{i.h.}{\equiv}$ $y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle (C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\})]$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}$,
with $C\{\ \} = y{<}^{\widehat{y_1}}_{\widehat{y_2}}\langle C'\{\ \}]$

5. $P\widehat{\alpha} \dagger \widehat{x}([M\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta)$
$\overset{cc_t4}{\equiv}$ $[(P\widehat{\alpha} \dagger \widehat{x}M)\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}b$
$\overset{i.h.}{\equiv}$ $[(C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\})\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta$
$\triangleq$ $C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}$,
with $C\{\ \} = [C'\{\ \}\rangle^{\widehat{\beta_1}}_{\widehat{\beta_2}}{>}\beta$

$$\begin{array}{lll}
6.\ P\widehat{\alpha} \dagger \widehat{x}(y \odot M) & \overset{wg1}{\equiv} & y \odot (P\widehat{\alpha} \dagger \widehat{x}M) \\
& \overset{i.h.}{\equiv} & y \odot (C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\}) \\
& \triangleq & C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}, \\
& & \quad \text{with} \ \ C\{\ \} = y \odot (C'\{\ \}) \\
7.\ P\widehat{\alpha} \dagger \widehat{x}(M \odot \beta) & \overset{wg2}{\equiv} & (P\widehat{\alpha} \dagger \widehat{x}M) \odot \beta \\
& \overset{i.h.}{\equiv} & (C'\{P\widehat{\alpha} \dagger \widehat{x}M^x\}) \odot \beta \\
& \triangleq & C\{P\widehat{\alpha} \dagger \widehat{x}M^x\}, \\
& & \quad \text{with} \ \ C\{\ \} = (C'\{\ \}) \odot \beta
\end{array}$$

Thus we are done with the proof of the second lemma. □

The following theorem confirms that the reduction procedure is complete. It shows that, if arbitrary term contains a cut, then there exists a representative of its congruence class which can be treated by some reduction rule.

The following theorem also shows that in the presence of congruence rules we do not need propagation rules. Propagation rules in the ${}^*\mathcal{X}$ calculus represent another approach to perform restructuring of the terms. However, thecongruence rules are more general in a sense that, unlike the propagation rules, they also define restructuring of normal forms (that is, the terms which do not contain cut).

**Theorem 68** *For every term of the form $P\widehat{\alpha} \dagger \widehat{x}Q$ there exists a context $C$ and terms $P^\alpha$ and $Q^x$ whose principal names are $\alpha, x$ respectively. and $P^\alpha \preccurlyeq P$, $Q^x \preccurlyeq Q$, such that*

$$P\widehat{\alpha} \dagger \widehat{x}Q \equiv C\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q^x\}$$

**Proof:** By using the two previous lemmas, we can construct the proof in two symmetric ways. This illustrates in practice our point of view about symmetry and equivalence of classical proofs.

$$\begin{array}{lll}
P\widehat{\alpha} \dagger \widehat{x}Q & \overset{lemma\ 66}{\equiv} & C_1\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q\} \\
& \overset{lemma\ 67}{\equiv} & C_1\{C_2\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q^x\}\} \\
& \triangleq & C'\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q^x\}, \quad \text{with} \quad C'\{\ \} = C_1\{C_2\{\ \}\}
\end{array}$$

The other way would be:

$$
\begin{array}{lll}
P\widehat{\alpha} \dagger \widehat{x}Q & \overset{lemma\ 67}{\equiv} & C_2\{P\widehat{\alpha} \dagger \widehat{x}Q^x\} \\
 & \overset{lemma\ 66}{\equiv} & C_2\{C_1\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q^x\}\} \\
 & \triangleq & C''\{P^\alpha\widehat{\alpha} \dagger \widehat{x}Q^x\}, \quad \text{with} \quad C''\{\ \} = C_2\{C_1\{\ \}\}
\end{array}
$$

And thus we are done. □

## 11.4 The reduction rules

The reduction relation in $^{\copyright}\mathcal{X}$, denoted by $\rightarrow$, is generated by the reduction rules (presented in Figures 5.5 , 5.7, 11.2 and 11.3) modulo the congruence relation (presented in Section 11.2).

The reduction rules are closely related to those of $^{*}\mathcal{X}$. Clearly, as we introduced the congruence relation to come closer to the diagrammatic framework, there is no need for propagation rules. This is proven in the previous section. Thus reduction system of the $^{\copyright}\mathcal{X}$ calculus includes *activation rules*, *deactivation rules*, *logical actions* and *structural actions*.

**Structural and logical actions** The formulation of structural and logical actions is identical to that given in Figures 5.5 and 5.7 on pages 52 and 53, respectively.

**Activation and deactivation rules** These rules in $^{\copyright}\mathcal{X}$ and $^{*}\mathcal{X}$ are principally the same. The difference is that activations in $^{\copyright}\mathcal{X}$ refer to more specific cases, as the cut-names are always introduced. Activation and deactivation is defined in Figures 11.2 and 11.3, respectively.

<u>*Left*</u> :

$$
\begin{array}{lll}
(act\text{-}L\text{-}eras) & : & (P \odot \alpha)\widehat{\alpha} \dagger \widehat{x}Q \rightarrow (P \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}Q \\
(act\text{-}L\text{-}dupl) & : & ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}} > \alpha)\widehat{\alpha} \dagger \widehat{x}Q \rightarrow ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}} > \alpha)\widehat{\alpha} \nearrow \widehat{x}Q
\end{array}
$$

<u>*Right*</u> :

$$
\begin{array}{lll}
(act\text{-}R\text{-}eras) & : & P\widehat{\alpha} \dagger \widehat{x}(x \odot Q) \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}(x \odot Q) \\
(act\text{-}R\text{-}dupl) & : & P\widehat{\alpha} \dagger \widehat{x}(x <^{\widehat{x_1}}_{\widehat{x_2}} \langle Q]) \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}(x <^{\widehat{x_1}}_{\widehat{x_2}} \langle Q])
\end{array}
$$

Figure 11.2: Activation rules for $^{\copyright}\mathcal{X}$

$$
\begin{array}{lll}
\underline{\textit{Left}}: & & \\
(\textit{cap}\nearrow\textit{-deact}) & : \langle x.\beta\rangle\widehat{\beta} \nearrow \widehat{y}R & \rightarrow \langle x.\beta\rangle\widehat{\beta} \dagger \widehat{y}R \\
(\textit{exp}\nearrow\textit{-deact}) & : (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \nearrow \widehat{y}R & \rightarrow (\widehat{x}\,P\,\widehat{\gamma}\cdot\beta)\widehat{\beta} \dagger \widehat{y}R \\
 & & \\
\underline{\textit{Right}}: & & \\
(\nwarrow\textit{cap-deact}) & : P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle & \rightarrow P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \\
(\nwarrow\textit{imp-deact}) & : P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\;[x]\;\widehat{y}\,R) & \rightarrow P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\;[x]\;\widehat{y}\,R)
\end{array}
$$

Figure 11.3: Deactivation rules for ${}^{\copyright}\mathcal{X}$

There exists a duality between activation and deactivation groups. If we formulate the activation and the deactivation rules in the following way

$$
\begin{array}{lll}
\underline{\textit{Activation}}: & & \\
(\textit{act-L}) & : P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nearrow \widehat{x}Q, & \alpha \text{ is S-principal for } P \\
(\textit{act-R}) & : P\widehat{\alpha} \dagger \widehat{x}Q \rightarrow P\widehat{\alpha} \nwarrow \widehat{x}Q, & x \text{ is S-principal for } X \\
 & & \\
\underline{\textit{Deactivation}}: & & \\
(\nearrow\textit{-deact}) & : P\widehat{\alpha} \nearrow \widehat{x}Q \rightarrow P\widehat{\alpha} \dagger \widehat{x}Q, & \alpha \text{ is L-principal for } P \\
(\nwarrow\textit{-deact}) & : P\widehat{\alpha} \nwarrow \widehat{x}Q \rightarrow P\widehat{\alpha} \dagger \widehat{x}Q, & x \text{ is L-principal for } X
\end{array}
$$

we may notice that by negating the conditions and reversing the arrow we go from left and right-activation to left and right-deactivation, respectively, and vice versa.

**Example 69** *Let us now revisit in the framework of* ${}^{\copyright}\mathcal{X}$ *Example 64 (given on page 136). The computation goes as follows:*

$$
\begin{array}{lll}
 & (\widehat{x}\,(\underline{\langle x.\alpha\rangle} \odot \beta)\,\widehat{\beta}\cdot\gamma)\widehat{\alpha} \dagger \widehat{y}(\widehat{z}\,(z \odot \underline{\langle y.\delta\rangle})\,\widehat{\delta}\cdot\eta) & \\
\equiv & \widehat{x}\,(\widehat{z}\,(z \odot (\underline{\langle x.\alpha\rangle\widehat{\alpha} \dagger \widehat{y}\langle y.\delta\rangle}) \odot \beta)\,\widehat{\delta}\cdot\eta)\,\widehat{\beta}\cdot\gamma & \\
\rightarrow & \widehat{x}\,(\widehat{z}\,(z \odot \underline{\langle x.\delta\rangle} \odot \beta)\,\widehat{\delta}\cdot\eta)\,\widehat{\beta}\cdot\gamma &
\end{array}
$$

This one-dimensional computation closely mirrors the corresponding diagrammatic reduction (shown in Example 65 on page 136), and that is what

we wanted to achieve. Indeed, reduction rules (modulo congruence rules) in ©$\mathcal{X}$ correspond to reduction rules of $^{d}\mathcal{X}$. This is formally proven in Section 11.8.

## 11.5 Operational properties

The congruence and reduction relations enjoy the properties of free names preservation and linearity preservation.

**Theorem 70 (Basic properties of $\equiv$ and $\rightarrow$)**

1. *Preservation of* free names *(interface preservation):*
   - *If* $P \equiv Q$ *then* $N(P) = N(Q)$
   - *If* $P \rightarrow Q$ *then* $N(P) = N(Q)$
2. *Preservation of* linearity*:*
   - *If* $P$ *is linear and* $P \equiv Q$ *then* $Q$ *is linear*
   - *If* $P$ *is linear and* $P \rightarrow Q$ *then* $Q$ *is linear*

**Proof:** They can be checked by treating carefully each rule. □

## 11.6 The typed language

The type system for ©$\mathcal{X}$ calculus is the same as for $^{*}\mathcal{X}$ calculus, and it is presented in Figure 5.1 on page 38.

### Type preservation

Besides the fact that these two calculi have the same syntax, the set of reduction rules of ©$\mathcal{X}$ is a subset of that of $^{*}\mathcal{X}$. Therefore it is assured by Theorem 31 that the reduction rules preserve types. There is however the need to prove that congruence rules preserve types.

**Theorem 71 ($\equiv$ preserves types)** *Let* $S$ *be an* ©$\mathcal{X}$*-term and* $\Gamma, \Delta$ *contexts. Then the following holds:*

$$\textit{If } S :\cdot\ \Gamma \vdash \Delta \textit{ and } S \equiv S', \textit{ then } S' :\cdot\ \Gamma \vdash \Delta$$

**Proof:** The proof goes by checking that the property holds for all equations defining $\equiv$. We first write the typing derivation for the term on the left-hand side, and then for the term on the right-hand side of the equation. We prove the cases which do not involve cuts and which are not tivial. As for the cases with cuts, the analysis is similar as for propagation rules of ${}^*\mathcal{X}$ (see Theorem 31 on page 63).

• Observe the (exp-exp) group.
– The only congruence rule is: $\widehat{y}\,(\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\cdot\delta \overset{ee}{\equiv} \widehat{x}\,(\widehat{y}\,P\,\widehat{\gamma}\cdot\delta)\,\widehat{\beta}\cdot\alpha$, with $\alpha\neq\gamma,\ \beta\neq\delta$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \vdots\ \Gamma, x:A, y:C \vdash \beta:B, \gamma:D, \Delta}{\boxed{\widehat{x}\,P\,\widehat{\beta}\cdot\alpha} \vdots\ \Gamma, y:C \vdash \gamma:D, \alpha:A\to B, \Delta}(\to R)}{\boxed{\widehat{y}\,(\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\cdot\delta} \vdots\ \Gamma \vdash \alpha:A\to B, \delta:C\to D, \Delta}(\to R)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} \vdots\ \Gamma, x:A, y:C \vdash \beta:B, \gamma:D, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\gamma}\cdot\delta} \vdots\ \Gamma, x:A \vdash \beta:B, \delta:C\to D, \Delta}(\to R)}{\boxed{\widehat{x}\,(\widehat{y}\,P\,\widehat{\gamma}\cdot\delta)\,\widehat{\beta}\cdot\alpha} \vdots\ \Gamma \vdash \alpha:A\to B, \delta:C\to D, \Delta}(\to R)$$

• Observe the (exp-imp) group.
– Take the <u>first</u> rule: $\widehat{x}\,(P\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \overset{ei1}{\equiv} (\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q$, with <u>$x,\beta\in N(P)$</u>, $x\neq z$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \vdots\ \Gamma, x:A \vdash \beta:B, \gamma:C, \Delta \qquad \boxed{Q} \vdots\ \Gamma', y:D \vdash \Delta'}{\boxed{P\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q} \vdots\ \Gamma, \Gamma', x:A, z:C\to D \vdash \beta:B, \Delta, \Delta'}(\to L)}{\boxed{\widehat{x}\,(P\,\widehat{\gamma}\ [z]\ \widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha} \vdots\ \Gamma, \Gamma', z:C\to D \vdash \alpha:A\to B, \Delta, \Delta'}(\to R)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} \,:\cdot\, \Gamma, x:A \vdash \beta:B, \gamma:C, \Delta}{\boxed{\widehat{x}\,P\,\widehat{\beta}\cdot\alpha} \,:\cdot\, \Gamma \vdash \alpha:A\to B, \gamma:C, \Delta}(\to R) \qquad \boxed{Q} \,:\cdot\, \Gamma', y:D \vdash \Delta'}{\boxed{(\widehat{x}\,P\,\widehat{\beta}\cdot\alpha)\,\widehat{\gamma}\,[z]\,\widehat{y}\,Q} \,:\cdot\, \Gamma, \Gamma', z:C\to D \vdash \alpha:A\to B, \Delta, \Delta'}(\to L)$$

– Take the second rule: $\widehat{x}\,(P\,\widehat{\gamma}\,[z]\,\widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha \overset{ei2}{\equiv} P\,\widehat{\gamma}\,[z]\,\widehat{y}\,(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)$, with $x, \beta \in N(Q)$, $x \neq z$. On the one hand we have:

$$\dfrac{\boxed{P} \,:\cdot\, \Gamma \vdash \gamma:C, \Delta \qquad \boxed{Q} \,:\cdot\, \Gamma', x:A, y:D \vdash \beta:B, \Delta'}{\dfrac{\boxed{P\,\widehat{\gamma}\,[z]\,\widehat{y}\,Q} \,:\cdot\, \Gamma, \Gamma', x:A, z:C\to D \vdash \beta:B, \Delta, \Delta'}{\boxed{\widehat{x}\,(P\,\widehat{\gamma}\,[z]\,\widehat{y}\,Q)\,\widehat{\beta}\cdot\alpha} \,:\cdot\, \Gamma, \Gamma', z:C\to D \vdash \alpha:A\to B, \Delta, \Delta'}(\to R)}(\to L)$$

On the other hand,

$$\dfrac{\boxed{P} \,:\cdot\, \Gamma \vdash \gamma:C, \Delta \qquad \dfrac{\boxed{Q} \,:\cdot\, \Gamma', x:A, y:D \vdash \beta:B, \Delta'}{\boxed{\widehat{x}\,P\,\widehat{\beta}\cdot\alpha} \,:\cdot\, \Gamma', y:D \vdash \alpha:A\to B, \Delta'}(\to R)}{\boxed{P\,\widehat{\gamma}\,[z]\,\widehat{y}\,(\widehat{x}\,Q\,\widehat{\beta}\cdot\alpha)} \,:\cdot\, \Gamma, \Gamma', z:C\to D \vdash \alpha:A\to B, \Delta, \Delta'}(\to L)$$

• Observe the (imp-imp) group.

– Take the first rule: $(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R \overset{ii1}{\equiv} (P\,\widehat{\beta}\,[z]\,\widehat{t}\,R)\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q$, with $\alpha, \beta \in N(P)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \,:\cdot\, \Gamma \vdash \alpha:A, \beta:C, \Delta \qquad \boxed{Q} \,:\cdot\, \Gamma', y:B \vdash \Delta'}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q} \,:\cdot\, \Gamma, \Gamma', x:A\to B \vdash \beta:C, \Delta, \Delta'}(\to L) \qquad \boxed{R} \,:\cdot\, \Gamma'', t:D \vdash \Delta''}{\boxed{(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R} \,:\cdot\, \Gamma, \Gamma', \Gamma'', x:A\to B, z:C\to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \beta : C, \Delta \qquad \boxed{R} :\cdot\ \Gamma'', t : D \vdash \Delta''}{\boxed{P\,\widehat{\beta}\,[z]\,\widehat{t}\,R} :\cdot\ \Gamma, \Gamma'', z : C \to D \vdash \alpha : A, \Delta, \Delta''}(\to L) \qquad \boxed{Q} :\cdot\ \Gamma', y : B \vdash \Delta'}{\boxed{(P\,\widehat{\beta}\,[z]\,\widehat{t}\,R)\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B, z : C \to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

– Take the second rule: $(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R \overset{ii2}{\equiv} P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R)$, with $y, \beta \in N(Q)$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \boxed{Q} :\cdot\ \Gamma', y : B \vdash \beta : C, \Delta'}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q} :\cdot\ \Gamma, \Gamma', x : A \to B \vdash \beta : C, \Delta, \Delta'}(\to L) \qquad \boxed{R} :\cdot\ \Gamma'', t : D \vdash \Delta''}{\boxed{(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,Q)\,\widehat{\beta}\,[z]\,\widehat{t}\,R} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B, z : C \to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

On the other hand,

$$\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{\boxed{Q} :\cdot\ \Gamma', y : B \vdash \beta : C, \Delta' \qquad \boxed{R} :\cdot\ \Gamma'', t : D \vdash \Delta''}{\boxed{Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R} :\cdot\ \Gamma, \Gamma', y : B, z : C \to D \vdash \Delta, \Delta'}(\to L)}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R)} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B, z : C \to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

– Take the third rule: $P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R) \overset{ii3}{\equiv} Q\,\widehat{\beta}\,[z]\,\widehat{t}\,(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,R)$, with $y, t \in N(R)$. On the one hand we have:

$$\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \dfrac{\boxed{Q} :\cdot\ \Gamma' \vdash \beta : C, \Delta' \qquad \boxed{R} :\cdot\ \Gamma'', y : B, t : D \vdash \Delta''}{\boxed{Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R} :\cdot\ \Gamma, \Gamma', y : B, z : C \to D \vdash \Delta, \Delta'}(\to L)}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{y}\,(Q\,\widehat{\beta}\,[z]\,\widehat{t}\,R)} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B, z : C \to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} :\cdot\ \Gamma \vdash \alpha : A, \Delta \qquad \boxed{R} :\cdot\ \Gamma'', y : B, t : D \vdash \Delta''}{\boxed{P\,\widehat{\alpha}\,[x]\,\widehat{y}\,R} :\cdot\ \Gamma, \Gamma'', x : A \to B, t : D \vdash \Delta, \Delta'}(\to L) \qquad \boxed{Q} :\cdot\ \Gamma' \vdash \beta : C, \Delta'}{\boxed{Q\,\widehat{\beta}\,[z]\,\widehat{t}\,(P\,\widehat{\alpha}\,[x]\,\widehat{y}\,R)} :\cdot\ \Gamma, \Gamma', \Gamma'', x : A \to B, z : C \to D \vdash \Delta, \Delta', \Delta''}(\to L)$$

• Observe the (exp-cont) group.

– Take the first rule: $x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle\widehat{y}\,P\,\widehat{\beta}\cdot\gamma]\ \overset{ec_t1}{\equiv}\ \widehat{y}\,(x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P])\,\widehat{\beta}\cdot\gamma$, with $y \neq x$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \because\ \Gamma, x_1 : A, x_2 : A, y : B \vdash\ \beta : C, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\gamma} \because\ \Gamma, x_1 : A, x_2 : A \vdash\ \gamma : B \to C, \Delta}\,(\to R)}{\boxed{x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle\widehat{y}\,P\,\widehat{\beta}\cdot\gamma]} \because\ \Gamma, x : A \vdash\ \gamma : B \to C, \Delta}\,(\textit{cont-L})$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} \because\ \Gamma, x_1 : A, x_2 : A, y : B \vdash\ \beta : C, \Delta}{\boxed{x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P]} \because\ \Gamma, x : A, y : B \vdash\ \beta : C, \Delta}\,(\textit{cont-L})}{\boxed{\widehat{y}\,(x{<}\frac{\widehat{x_1}}{\widehat{x_2}}\langle P])\,\widehat{\beta}\cdot\gamma} \because\ \Gamma, x : A \vdash\ \gamma : B \to C, \Delta}\,(\to R)$$

– Take the second rule: $[\widehat{y}\,P\,\widehat{\beta}\cdot\gamma\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}}{>}\alpha\ \overset{ec_t2}{\equiv}\ \widehat{y}\,([P\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}}{>}\alpha)\,\widehat{\beta}\cdot\gamma$, with $\alpha \neq \beta$, $\alpha_1, \alpha_2 \neq \gamma$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \because\ \Gamma, y : B \vdash\ \alpha_1 : A, \alpha_2 : A, \beta : C, \Delta}{\boxed{\widehat{y}\,P\,\widehat{\beta}\cdot\gamma} \because\ \Gamma \vdash\ \alpha_1 : A, \alpha_2 : A, \gamma : B \to C, \Delta}\,(\to R)}{\boxed{[\widehat{y}\,P\,\widehat{\beta}\cdot\gamma\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}}{>}\alpha} \because\ \Gamma \vdash\ \alpha : A, \gamma : B \to C, \Delta}\,(\textit{cont-R})$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} \because\ \Gamma, y : B \vdash\ \alpha_1 : A, \alpha_2 : A, \beta : C, \Delta}{\boxed{[P\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}}{>}\alpha} \because\ \Gamma, y : B \vdash\ \alpha : A, \beta : C, \Delta}\,(\textit{cont-R})}{\boxed{\widehat{y}\,([P\rangle_{\widehat{\alpha_2}}^{\widehat{\alpha_1}}{>}\alpha)\,\widehat{\beta}\cdot\gamma} \because\ \Gamma \vdash\ \alpha : A, \gamma : B \to C, \Delta}\,(\to R)$$

• In the case of (imp-cont) group the proofs goes similarly to the previous.

• Observe the (cont-assoc) group.
– Take the <u>first</u> rule: $z<\widehat{\frac{y}{x_3}}\langle y<\widehat{\frac{x_1}{x_2}}\langle P]]\ \overset{c_tas1}{\equiv}\ z<\widehat{\frac{x_1}{y}}\langle y<\widehat{\frac{x_2}{x_3}}\langle P]]$. On the one hand we have:

$$\dfrac{\dfrac{\boxed{P} \vdots\ \Gamma, x_1 : A, x_2 : A, x_3 : A \vdash \Delta}{\boxed{y<\widehat{\frac{x_1}{x_2}}\langle P]} \vdots\ \Gamma, y : A, x_3 : A \vdash \Delta}\,(\textit{cont-L})}{\boxed{z<\widehat{\frac{y}{x_3}}\langle y<\widehat{\frac{x_1}{x_2}}\langle P]]} \vdots\ \Gamma, z : A \vdash \Delta}\,(\textit{cont-L})$$

On the other hand,

$$\dfrac{\dfrac{\boxed{P} \vdots\ \Gamma, x_1 : A, x_2 : A, x_3 : A \vdash \Delta}{\boxed{y<\widehat{\frac{x_2}{x_3}}\langle P]} \vdots\ \Gamma, x_1 : A, y : A \vdash \Delta}\,(\textit{cont-L})}{\boxed{z<\widehat{\frac{x_1}{y}}\langle y<\widehat{\frac{x_2}{x_3}}\langle P]]} \vdots\ \Gamma, z : A \vdash \Delta}\,(\textit{cont-L})$$

– As for the <u>second</u> rule: $[[P\rangle\widehat{\frac{\alpha_1}{\alpha_2}}>\beta\rangle\widehat{\frac{\beta}{\alpha_3}}>\gamma\ \overset{c_tas2}{\equiv}\ [[P\rangle\widehat{\frac{\alpha_2}{\alpha_3}}>\beta\rangle\widehat{\frac{\alpha_1}{\beta}}>\gamma$, a similar proof to the previous case can be given. □ This section presents the relation between ${}^{\copyright}\mathcal{X}$ and ${}^{d}\mathcal{X}$ in a formal way.

## 11.7 Interpreting terms as diagrams

${}^{\copyright}\mathcal{X}$ calculus has been designed to stand in full correspondence with the diagrammatic calculus. Therefore it is not a surprise that we can define a simple translation from ${}^{\copyright}\mathcal{X}$-terms into ${}^{d}\mathcal{X}$-diagrams. This translation is denoted by $\mathcal{D}$ and it is defined inductively on the structure of terms, as presented in Figure 11.4. Similar approach was taken in [KL05, KL07], when interpreting λlxr-terms into proof-nets.

## 11.8 Simulating ${}^{\copyright}\mathcal{X}$-reduction

The following properties are satisfied by the translation $\mathcal{D}$.

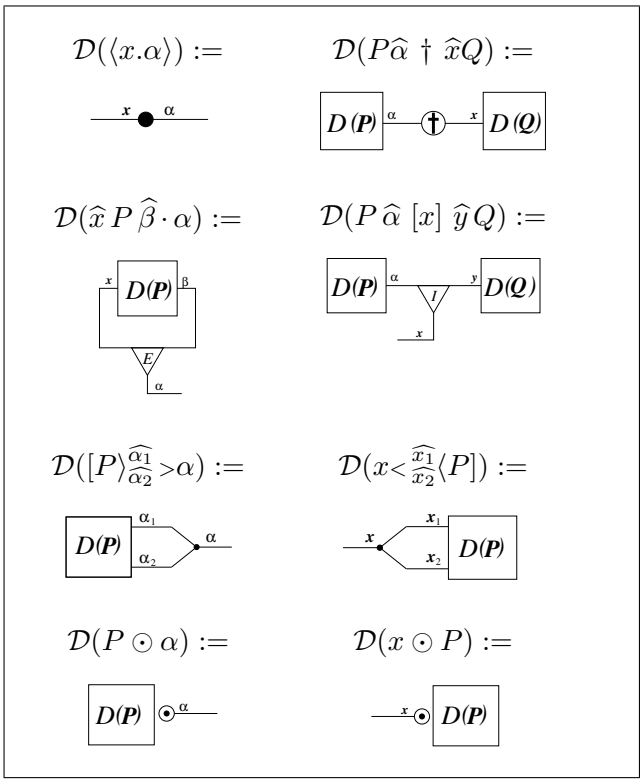



Figure 11.4: Encoding the terms into diagrams

**Theorem 72 (Simulation of reduction in ©$\mathcal{X}$)** *Let $P$ and $Q$ be ©$\mathcal{X}$ terms. Then the following holds:*

- $P_1 \equiv P_2$ *then* $\mathcal{D}(P_1) = \mathcal{D}(P_2)$
- $P_1 \rightarrow P_2$ *then* $\mathcal{D}(P_1) \rightarrow \mathcal{D}(P_2)$

**Proof:** The first part is satisfied since the congruence rules are introduced to identify terms who share the same diagrammatic representation. The second part of the proof goes by analyzing all reduction rules.

**Activation rules:**

- For the *(actL-eras)* rule: $(P \odot \alpha)\widehat{\alpha} \dagger \widehat{x}Q \;\rightarrow\; (P \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}Q$, we have:

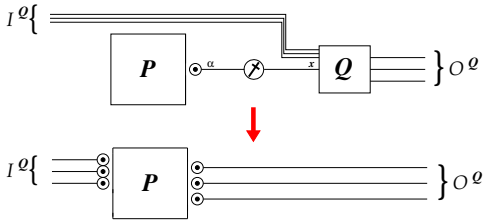
• For the *(actL-dupl)* rule: $([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\alpha} \dagger \widehat{x}Q \;\rightarrow\; ([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\alpha} \nearrow \widehat{x}Q$, we have:


• For the *(actR-eras)* rule: $P\widehat{\alpha} \dagger \widehat{x}(x \odot Q) \;\rightarrow\; P\widehat{\alpha} \nwarrow \widehat{x}(x \odot Q)$, we have:

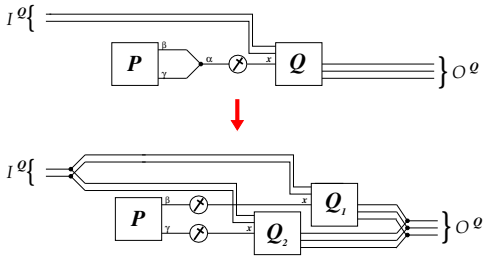

• For the *(actR-dupl)* rule: $P\widehat{\alpha} \dagger \widehat{x}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}(Q]) \;\rightarrow\; P\widehat{\alpha} \nwarrow \widehat{x}(x{<}^{\widehat{x_1}}_{\widehat{x_2}}(Q])$, we have:


**Structural actions:**

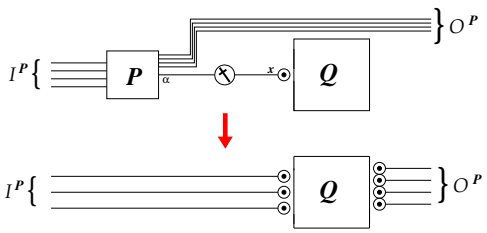
• For the *(↗-eras)* rule: $(P \odot \alpha)\widehat{\alpha} \nearrow \widehat{x}Q \;\rightarrow\; \mathcal{I}^Q \odot P \odot \mathcal{O}^Q$, with $\mathcal{I}^Q = \overline{I}(Q) \setminus x$, $\mathcal{O}^Q = \overline{O}(Q)$, we have:



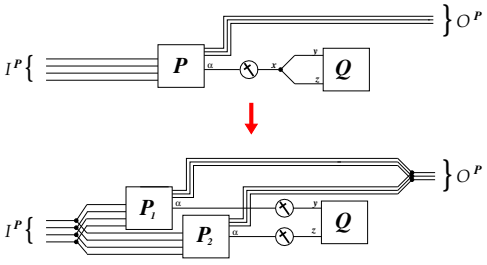

• For the *(↗-dupl)* rule: $([P\rangle^{\widehat{\alpha_1}}_{\widehat{\alpha_2}}{>}\alpha)\widehat{\alpha} \nearrow \widehat{x}Q \;\rightarrow\; \mathcal{I}^Q{<}^{\widehat{\mathcal{I}^Q_1}}_{\widehat{\mathcal{I}^Q_2}}\left\langle (P\widehat{\alpha_1} \nearrow \widehat{x_1}Q_1)\widehat{\alpha_2} \nearrow \widehat{x_2}Q_2 \right\rangle {}^{\widehat{\mathcal{O}^Q_1}}_{\widehat{\mathcal{O}^Q_2}}{>}\mathcal{O}^Q$, with $\mathcal{I}^Q = \overline{I}(Q) \setminus x$, $\mathcal{O}^Q = \overline{O}(Q)$, we have:




• For the *(↖-eras)* rule: $P\widehat{\alpha} \nwarrow \widehat{x}(x \odot Q) \;\rightarrow\; \mathcal{I}^Q \odot P \odot \mathcal{O}^Q$, with $\mathcal{I}^P = \bar{I}(P)$, $\mathcal{O}^P = \bar{O}(P) \setminus \alpha$, we have:




• For the *(↖-dupl)* rule: $P\widehat{\alpha} \nwarrow \widehat{x}(x{<}\widehat{x_1}\widehat{x_2}\langle Q\rangle) \;\rightarrow\; \mathcal{I}^P{<}\begin{matrix}\widehat{\mathcal{I}_1^P}\\ \widehat{\mathcal{I}_2^P}\end{matrix}\Big\langle P_2\widehat{\alpha_2} \nwarrow \widehat{x_2}(P_1\widehat{\alpha_1} \nwarrow \widehat{x_1}Q)\Big\rangle\begin{matrix}\widehat{\mathcal{O}_1^P}\\ \widehat{\mathcal{O}_2^P}\end{matrix}{>}\mathcal{O}^P$, with $\mathcal{I}^P = \bar{I}(P)$, $\mathcal{O}^P = \bar{O}(P) \setminus \alpha$, we have:




**Deactivation rules:**

• For the *(cap↗-deact)* rule: $\langle y.\alpha\rangle\widehat{\alpha} \nearrow \widehat{x}Q \;\rightarrow\; \langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}Q$, we have:

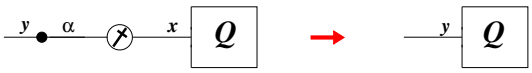

• For the *(exp↗-deact)* rule: $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}Q \;\rightarrow\; (\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha} \nearrow \widehat{x}Q$, we have:


• For the *(↖cap-deact)* rule: $P\widehat{\alpha} \nwarrow \widehat{x}\langle x.\beta\rangle \;\rightarrow\; P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle$, we have:

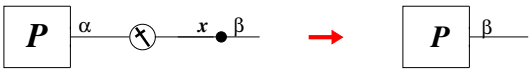

• For the *(↖imp-deact)* rule: $P\widehat{\alpha} \nwarrow \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R) \;\rightarrow\; P\widehat{\alpha} \dagger \widehat{x}(Q\,\widehat{\beta}\,[x]\,\widehat{y}\,R)$, we have:

**Logical actions:**

- For the *(ren-L)* rule: $\langle y.\alpha\rangle\widehat{\alpha} \dagger \widehat{x}Q \rightarrow Q\{y/x\}$, we have:

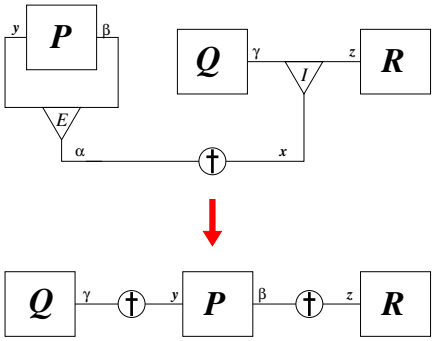



- For the *(exp-ren)* rule: $P\widehat{\alpha} \dagger \widehat{x}\langle x.\beta\rangle \rightarrow P\{\beta/\alpha\}$, we have:




- For the *(ei-insert)* rule: $(\widehat{y}\,P\,\widehat{\beta}\cdot\alpha)\widehat{\alpha}\dagger\widehat{x}(Q\,\widehat{\gamma}\,[x]\,\widehat{z}\,R) \rightarrow$ *either* $\begin{cases}(Q\widehat{\gamma}\dagger\widehat{y}P)\widehat{\beta}\dagger\widehat{z}R\\ Q\widehat{\gamma}\dagger\widehat{y}(P\widehat{\beta}\dagger\widehat{z}R)\end{cases}$

we have:




Thus we are done with the proof. □

# Conclusion and future work

This dissertation presents three formal languages through which a computational interpretation of classical logic was studied. The core of this work and the source of inspiration was the ${}^{*}\mathcal{X}$ calculus, presented in the first part and created by applying to the $\mathcal{X}$ calculus ideas at the origin of the intuitionistic $\lambda\mathsf{lxr}$ calculus. This was done smoothly and naturally in such a way that essential properties were preserved.

The ${}^{*}\mathcal{X}$ calculus is a finest grain computational language, which stands in a direct Curry-Howard correspondence with classical logic. It is a tool which allows us to study the role of structural rules in proof-transformation. On the computational side, this means that it provides us with an insight of how erasure and duplication can be implemented, in contrast with computational models which have implicit erasure and duplication, and are therefore distant from the actual implementations. On the one hand, we have a high level of details carried by the syntax, but on the other hand, the fact that everything is revealed lead to a design of a more abstract model, namely the diagrammatic calculus for classical logic.

Instead of focusing only on ${}^{*}\mathcal{X}$ calculus, we decided to present the other two models derived from it, namely the ${}^{d}\mathcal{X}$ and the ${}^{\copyright}\mathcal{X}$ calculus. This allowed us to relate the different approaches reflected by these three calculi. As a result we reveal both, the part which can be considered the essential part of classical computation, but also the part which represents unessential details (originating from the sequent calculus), and how these details are manipulated in the computation.

The ${}^{d}\mathcal{X}$ calculus is a diagrammatic calculus for classical logic, whose diagrams are derived from ${}^{*}\mathcal{X}$-terms. It captures the essence of the classical computation, while abstracting away from unessential details.

The ${}^{\copyright}\mathcal{X}$ calculus is a one-dimensional counterpart of the diagrammatic calculus, and thus it makes a clear distinction between essential and unessential parts - in the one-dimensional framework. Essential part is represented by reduction rules, whilst the unessential part is given by the congruence

rules. ${}^{\copyright}\mathcal{X}$ is obtained by deepening the understanding of the relation between the first two calculi, namely ${}^{*}\mathcal{X}$ and ${}^{d}\mathcal{X}$. By introducing the congruence relation we identify the ${}^{*}\mathcal{X}$-terms which have the same diagrammatic representation, and which can be considered as computationally equivalent. Reduction rules are defined modulo congruence rules.

In some sense this thesis is a journey from one-dimensional to two-dimensional classical computation, and back. It represents a step towards revisiting classical proof theory in a framework where proofs are identified up to trivial rule permutations. It also suggests that $\mathcal{X}$-like calculi, that is, the calculi which code the structure of classical proofs using the syntax similar to that of $\mathcal{X}$ calculus [vBLL05, vBL07], should be considered as a relevant tool for the study of different aspects of classical computation.

**Future works** There are several areas where it is possible to continue the work presented in this thesis.

- An interesting direction for future work is to reconsider the one and two-dimensional classical computation in the three-dimensional framework. The three-dimensional proofs have been studied in [Gui06], where calculus of structures is an underlying formalism.

- Exchange rule is the third structural rule. What is the role of this rule in the computation? Clearly, the term-language which would take into consideration all three structural rules (exchange, besides weakening and contraction) would be even more complex syntactically, but one should investigate how this would affect the two-dimensional variant of that calculus.

- We have not considered the strong normalization issues. The ${}^{*}\mathcal{X}$ calculus has been derived from $\mathcal{X}$ calculus for which the strong normalization is proven in [UB01a]. We expect that this property is not lost, and that it is preserved not only in ${}^{*}\mathcal{X}$, but also in ${}^{d}\mathcal{X}$ and ${}^{\copyright}\mathcal{X}$.

- The diagrammatic calculus, ${}^{d}\mathcal{X}$, exhibits several sources of non-determinism. Some of them are known to lead to non-confluence, but for the others it should be formally investigated.

# Bibliography


[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[AriBC] Aristotle. Organon, circa 350 BC.

[AS03] B. André-Salvini. *Le code de Hammurabi*. Musée du Louvre, Paris, 2003. Collection *Solo*, n. 27.

[AvB07] P. Audebaud and S. van Bakel. A completeness result for $\lambda_\mu$. Preprint, 2007.

[Bar84] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

[BB94] F. Barbanera and S. Berardi. A symmetric lambda calculus for "classical" program extraction. In *TACS*, pages 495–515, 1994.

[BBLRD96] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.

[BBS97] F. Barbanera, S. Berardi, and M. Schivalocchi. "Classical" programming-with-proofs in $\lambda^{sym}$: an analysis of non-confluence. In *TACS*, pages 365–390, 1997.

[BR95] R. Bloo and K.H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN'95 Computer Science in the Netherlands*, pages 62–72, 1995.

[Bus91] S. R. Buss. The undecidability of k-provability. *Annals of Pure and Applied Logic*, 53(1):75–102, 1991.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland Publishing Co., Amsterdam, 1958.

[CH00] P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. 5 th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP'00)*, pages 233–243. ACM, 2000.

[Chu85] A. Church. *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*. Princeton University Press, Princeton, NJ, USA, 1985.

[CKP03] R. Di Cosmo, D. Kesner, and E. Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Computer Science*, 13(3):409–450, 2003.

[dG94] P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In *5th Int. Conf. LPAR*, volume 822 of *Lecture Notes in Computer Science*, pages 31–43, 1994.

[DG01] R. David and B. Guillaume. A lambda-calculus with explicit weakening and explicit substitution. *Mathematical Structures in Computer Science*, 11(1):169–206, 2001.

[DJS96] V. Danos, J.-B. Joinet, and H. Schellinx. Computational isomorphisms in classical logic (extended abstract). *Electronic Notes in Theoretical Computer Science*, 3, 1996.

[DJS97] V. Danos, J.-B. Joinet, and H. Schellinx. A new deconstructive logic: Linear logic. *Journal of Symbolic Logic*, 62, 1997.

[Eul68] L. Euler. *Lettres à une Princesse d'Allemagne*. l'Académie Impériale des Sciences, St. Petersburg, 1768.

[Fil89] A. Filinski. Declarative continuations: an investigation of duality in programming language semantics. In *Category Theory and Computer Science*, pages 224–249, London, UK, 1989. Springer-Verlag.

[GAL92] G. Gonthier, M. Abadi, and J.-J. Lévy. Linear logic without boxes. In *LICS*, pages 223–234, 1992.

[Gen35] G. Gentzen. Untersuchungen über das logische Schließen. *Math. Z.*, 39:176–210, 405–431, 1935.

[Ghi87] B. Ghiselin, editor. *The creative process: Reflections on invention in the arts and sciences*. University of California Press, Berkeley, CA, 1987. (The original manuscript written in 1952).

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir91] J.-Y. Girard. A new constructive logic: classical logic. *Mathematical Stuctures in Computer Science*, 1(3):255–296, 1991.

[Gir01] J.-Y. Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoret Computer Science*. Cambridge University Press, 1989.

[Gri90] T. Griffin. A formulae-as-types notion of control. In *Proceedings of the 17th ACM symposium on Principles of programming languages, POPL*, pages 47–58, 1990.

[Gug04] A. Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 2004. (The manuscript was written in 1999).

[Gui04] Y. Guiraud. *Présentations d'opérades et systèmes de réécriture*. Thèse de doctorat, Université Montpellier II, Juin 2004.

[Gui06] Y. Guiraud. The three dimensions of proofs. *Annals of Pure and Applied Logic*, 141(1-2):266–295, 2006.

[Her95] H. Herbelin. *Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse de doctorat, Université Paris VII, Janvier 1995.

[Her01] H. Herbelin. Explicit substitution and reducibility. *Journal of Logic and Computation*, 11(3):431–451, 2001.

[Her05] H. Herbelin. *C'est maintenant qu'on calcule, au cœur de la dualité*. Habilitation à diriger les recherches, Université Paris XI, Décembre 2005.

[How80] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. (The unpublished manuscript is written in 1969).

[Hyl02] J. M. E. Hyland. Proof theory in the abstract. *Annals of Pure and Applied Logic*, 114(1-3):43–78, 2002.

[Joh82] B. Johnston, editor. *My Inventions: The Autobiography of Nikola Tesla*. Barnes and Noble, New York, 1982.

[Kes07] D. Kesner. The theory of calculi with explicit substitutions revisited. In *Proceedings of the 16th EACSL Annual Conference on Computer Science and Logic*, pages 238–252, September 2007.

[KL05] D. Kesner and S. Lengrand. Extending the explicit substitution paradigm. In *RTA*, pages 407–422, 2005.

[KL06] D. Kesner and S. Lengrand. Explicit operators for $\lambda$-calculus. *Information and Computation*, 2006. Extended version of a communication at RTA-05.

[KL07] D. Kesner and S. Lengrand. Ressource operators for lambda-calculus. *Information and Computation*, 205(4):419–473, 2007. Long version.

[Kle52] S. Kleene. *Introduction to Metamathematics*. Number 1 in Bibliotheca mathematica. North-Holland, 1952. Revised edition, Wolters-Noordhoff, 1971.

[Laf90] Y. Lafont. Interaction nets. In *Proceedings of the 17th ACM symposium on Principles of programming languages, POPL*, pages 95–108. ACM Press, 1990.

[Laf95a] Y. Lafont. Equational reasoning with 2-dimensional diagrams. In *Term Rewriting*, volume 909 of *Lecture Notes in Computer Science*, pages 170–195. Springer-Verlag, 1995.

[Laf95b] Y. Lafont. From proof-nets to interaction nets. In *Advances in linear logic*, pages 225–247. Cambridge University Press, 1995.

[Laf97] Y. Lafont. Interaction combinators. *Information and Computation*, 137(1):69–101, 1997.

[Len03] S. Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In *Electronic Notes in Theoretical Computer Science*, volume 86, 2003.

[Les94] P. Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In *Proceedings of the 21st ACM symposium on Principles of programming languages, POPL*, pages 60–69. ACM Press, 1994.

[LLD$^+$04] S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Information and Computation*, 189(1):17–42, 2004.

[LS05] F. Lamarche and L. Straßburger. Naming proofs in classical propositional logic. In *TLCA*, pages 246–261, 2005.

[Mel95] P.-A. Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In *TLCA*, pages 328–334, 1995.

[Mil95] R. Milner. *Communication and concurrency*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, 1995.

[OS97] C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proc.24th Annual ACM Symp. on Principles Of Programming Languages*, pages 215–227, 1997.

[Par92] M. Parigot. An algorithmic interpretation of classical natural deduction. In *Int. Conf. LPAR*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201, 1992.

[Par97] M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, December 1997.

[Pol04] E. Polonovski. Strong normalization of $\bar{\lambda}\mu\tilde{\mu}$-calculus with explicit substitutions. In *FoSSaCS*, pages 423–437, 2004.

[Rob03] E. Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13(5):777–797, 2003.

[RS07] J. Raghunandan and A. J. Summers. On the computational representation of classical logical connectives. *Electronic Notes in Theoretical Computer Science*, 171(3):85–109, 2007.

[Str06] L. Straßburger. Proof nets and the identity of proofs. *CoRR*, abs/cs/0610123, 2006.

[SvB06] A. J. Summers and S. van Bakel. Approaches to polymorphism in classical sequent calculus. In *ESOP*, pages 84–99, 2006.

[SW01] D. Sangiorgi and D. Walker. *$\pi$-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, USA, 2001.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Comuter Science*. Cambridge University Press, 2003.

[TS96] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, New York, NY, USA, 1996.

[UB01a] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundamenta Informaticae*, 45(1-2):123–155, 2001. (appeared also at *TLCA* in 1999).

[UB01b] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundam. Inf.*, 45(1,2):123–155, 2001.

[Urb00] C. Urban. *Classical Logic and Computation*. PhD thesis, Univ. of Cambridge, October 2000.

[Urb01] C. Urban. Strong normalisation for a Gentzen-like cut-elimination procedure. In *Typed Lambda Calculus and Applications*, volume 2044 of *Lecture Notes in Computer Science*, pages 415–429, 2001.

[vB05] S. van Bakel. Intersection and union types for $\mathcal{X}$. *Electronic Notes in Theoretical Computer Science*, 136:203–227, 2005.

[vBCV07] S. van Bakel, L. Cardelli, and M. G. Vigliotti. From $\mathcal{X}$ to $\pi$. Representing the classical sequent calculus in $\pi$-calculus. Preprint, 2007.

[vBL07] S. van Bakel and P. Lescanne. Computation with classical sequents. *Mathematical Stuctures in Computer Science*, 2007. To appear.

[vBLL05] S. van Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In *Proc.9th Italian Conf. on Theoretical Computer Science (ICTCS'05)*, volume 3701 of *Lecture Notes in Computer Science*, pages 81–96, 2005.

[vBR05] S. van Bakel and J. Raghunandan. Implementing $\mathcal{X}$. *Electronic Notes in Theoretical Computer Science*, 127(5):171–195, 2005.

[vH67] J. van Heijenoort, editor. *From Frege to Godel. A Source Book tn Mathematical Logic, 1879-1931.* Harward University Press, Cambridge, Massachusetts, USA, 1967.

[WR25] A. N. Whitehead and B. Russell. *Principia Mathematica.* Cambridge University Press, 2nd edition, 1925.

# List of Figures